VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Výkonnostní testování SIP infrastruktury

Performance testing of SIP infrastructure

2010 Jan Rozhon

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Jan Rozhon**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Výkonnostní testování SIP infrastruktury
Performance testing of SIP infrastructure

Zásady pro vypracování:

1. Otevřená řešení pro IP telefonii.
2. Generátor provozu na protokolu SIP a RTP.
3. Návrh testovacích scénářů.
4. Realizace zátěžových testů pro SIP Proxy a B2BUA.
5. Vyhodnocení statických a dynamických parametrů zatížení.

Seznam doporučené odborné literatury:

VOZŇÁK, M. *Voice over IP.* Ostrava: VŠB - TU Ostrava, 1. vydání, 2008. 176 s. ISBN 978-80-248-1828-3.

MEGGELEN, J., SMITH, J., MADSEN, L. *Asterisk: The Future of Telephony.* O'Reilly Media, 2nd Edition, 2007. ISBN 978-05-965-1048-0.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě, dne 7.5. 2010 …….....

Podpis

Poděkování

Velmi rád bych tímto poděkoval Doc. Ing. Miroslavu Vozňákovi, Ph.D, vedoucímu mé diplomové práce za trpělivé vedení a množství praktických rad. Dále Ing. Lukáši Macurovi, který mi pomohl svými cennými znalostmi a zkušenostmi.

ABSTRAKT

Práce se zabývá výkonnostním testováním SIP infrastruktury, vychází jednak z návrhů doporučení IETF a rovněž přináší vlastní metodiku testování. Aktuálnost tématu je dána faktem, že dosud neexistuje žádný standard. Metodika navržená v této diplomové práci je určena pro výkonnostní testování SIP serveru v konfiguraci SIP proxy i B2BUA, přičemž funkčnost této metodiky je okamžitě testována sadou několika měření. Výstupem těchto měření jsou data popisující dynamicky se měnící parametry SIP serveru, na jejichž základě je možné vyvodit jeho výkonnost. Analýza nasbíraných dat je integrální součástí této práce a čtenář je veden krok za krokem při jejich vyhodnocování s cílem definovat maximální přípustné zatížení SIP serveru. Hlavní výhodou praktického ověření navržené metodiky testovaní je to, že vychází čistě z open source programů a může být snadno reprodukovatelné.

KLÍČOVÁ SLOVA

Asterisk, B2BUA, Opensips, Sip Proxy, SIPp, výkonnostní testování

ABSTRACT

This thesis deals with the gap in the branch of SIP infrastructure performance testing, which is caused by nonexistence of international standards in this field. The way of performance testing of SIP server that has been designed as a part of this diploma thesis can be applied with both SIP Proxy and B2BUA operational mode. Achieved results have been proved by the series of measurements the output of which is the data describing the dynamically changing attributes of SIP server. The performance of the SIP Server can be determined by this methodology. The analysis of the collected data is a part of this thesis and whoever reads this work is guided through it step by step in order to recognize the maximum potential of SIP server. The main advantage of the design of the benchmarking tests is that they are completely based on open source solutions.

KEYWORDS

Asterisk, B2BUA, Opensips, Sip Proxy, SIPp, performance testing.

Seznam použitých zkratek

AEL	Asterisk Extension Language	Jazyk pro definování dialplanů v Asterisku
AWK	Aho, Weinberger, Kernighan	Jazyk pro zpracovávání textů
B2BUA	Back-to-Back User Agent	Varianta SIP serveru
BASH	Bourne again Shell	Unixový příkazový shell interpreter
BSD	Berkeley Software Distribution	Odvozenina Unixu univerzity Berkeley
CDR	Call Detail Record	Záznam detailů hovoru
CSV	Comma-separated Values	Formát textového souboru
DNS	Domain Name Server	Doménový jmenný server
DSA	Digital Signature Algorithm	Algoritmus digitálního šifrování
DUT	Device under Test	Testované zařízení
GNU	GNU's not Unix	Projekt zaměřený na svobodný software
GPL	General Public Licence	Všeobecná licence GNU
IAX	Inter-Asterisk Exchange	Signalizační protokol mezi ústřednami Asterisk
IETF	Internet Engineering Task Force	Komise techniky Internetu
ISDN	Integrated Service Digital Network	Digitální síť integrovaných služeb
IVR	Interactive Voice Response	Interaktivní hlasová odezva
PBX	Public Exchange	Pobočková ústředna
PCM	Pulse-code Modulation	Pulzní kódová modulace
PSTN	Public Switched Telephone Network	Analogová telefonní síť
RRD	Registration Request Delay	Délka výměny registračních zpráv
RSA	Rivest, Shamir, Adleman	Algoritmus digitálního šifrování
RTP	Real-time Transport Protocol	Protokol přenosu v reálném čase
SAR	Systém Activity Reporter	Nástroj pro sledování prostředků PC
SIP	Session Initiation Protocol	Signalizační protokol ve VoIP
SRD	Session Request Delay	Délka výstavby spojení
SSH	Secure Shell	Protokol pro zabezpečenou vzdálenou správu
SUT	Systém under Test	Testovaný systém (více zařízení)

UAC	User Agent Client	Klientská forma SIP koncového zařízení
UAS	User Agent Server	Serverová forma SIP koncového zařízení
XML	Extensible Markup Language	Rozšiřitelný značkovací jazyk

OBSAH

1 ÚVOD

S pokračujícím rozvojem Internetu procházejí i VoIP technologie bouřlivým vývojem. Je tomu zejména proto, že stále dostupnější širokopásmové připojení k Internetu umožňuje proniknout VoIP systémům i do oblastí, pro které se dříve vůbec nehodily, nebo pro které byly nepoužitelné. Dnes je tedy naprosto běžné, že firmy v rámci své interní sítě provozují některý z komunikačních systémů využívajících VoIP technologii, stejně jako je běžné, že domácnosti mají při výběru své konektivity do telefonní sítě na výběr i z připojení na bázi technologie VoIP.

Setkáváme se se stále větším počtem otevřených i proprietárních řešení využívajících VoIP technologii. IP telefonie představuje rostoucí konkurenci pro poskytovatele klasických telefonních služeb a to zejména z toho pohledu, že v době nejisté ekonomické situace jsou všechny subjekty (veřejné i soukromé) nuceny uvažovat nad způsoby, jak ušetřit, což dělá VoIP systémy o to atraktivnější, neboť cena je jejich hlavní devizou. V porovnání se zaběhnutými telekomunikačními operátory nabízejí VoIP poskytovatelé zlomkové ceny pro propojení do cizích sítí. Propojení v rámci sítě poskytovatele VoIP pak je obvykle pro zákazníka zcela zdarma, což skýtá zajímavý potenciál pro uplatnění na trhu.

Navzdory těmto optimistickým vizím a poměrně výhodným podmínkám, které VoIP systémy nabízejí, nejsou tyto systémy páteří moderní komunikace. Stále tvoří jen značně menšinový proud v komunikacích. Hlavní důvody pro tento stav jsou, že VoIP systémy jsou považovány za nespolehlivé (většinou neprávem) a následně pak povinnost velkých poskytovatelů zajišťovat veřejně dostupnou hlasovou službu, která poskytovatele vede k tomu, aby každou nově nasazovanou technologii podrobili velmi důkladnému prověřování a veškeré velké projekty pečlivě připravovali.

Navíc, pokud poskytovatelé VoIP služeb přijdou o svou hlavní výhodu – tedy nízkou cenu, jako se tomu událo na českém trhu v loňském roce (2009) u jednoho z větších poskytovatelů VoIP služeb (viz [1]), ztrácí tím i svou atraktivitu, neboť běžný zákazník další z výhod VoIP, mezi které patří například možnost více telefonních přístrojů s jedním číslem, popřípadě integrace dalších (zejména datových) služeb, většinou nevyužije.

S VoIP systémy tak vlastně zůstanou v očích potenciálních zákazníků spojeny pouze jejich nevýhody, z nichž je v podvědomí mnoha lidí nejhlouběji zapsána nespolehlivost a nízká kvalita hovoru. Přitom se ale jedná o problémy, které nejsou výlučným znakem VoIP telefonie – každá telefonní síť totiž zažívá výpadky. Navíc výkon moderní VoIP infrastruktury umožňuje bezproblémový provoz. Problémy totiž nejsou většinou spojeny přímo s poskytovatelem VoIP služeb, jako spíše s vlastním připojením na Internet.

Spolehlivost a možnosti VoIP systémů jsou tedy mezi zákazníky mnohdy podceňovány a i v rámci VoIP komunity neexistuje jednotná metodika, jak výkonnost a spolehlivost VoIP systémů určovat.

Toto je hlavní důvod, proč se v této práci budu snažit nastínit metodiku a způsoby, jak testovat infrastrukturu založenou na hlavním signalizačním protokolu, který je ve VoIP systémech využíván, tedy na protokolu SIP.

Teoretický základ k problematice VoIP systémů a SIPu, čtenář najde v příloze P I. Tato část má za úkol zasadit problematiku do teoretického rámce tak, aby čtenář této práce rozuměl termínům a obratům používaných v dalších kapitolách této práce a aby pochopil základní filozofii signalizačního protokolu SIP a jeho úlohy při telefonování prostřednictvím VoIP systémů. Rozebrán zde bude i konkurenční protokol H.323, aby si čtenář udělal ucelený obraz o současnosti signalizačních protokolů. Na závěr této části pak bude nastíněna role protokolu pro přenos médií – RTP. Tento text není umístěn v hlavní části této práce zejména z důvodu, že potenciální čtenáři této práce mají s problematikou SIPu a VoIP značné zkušenosti. Přesto pro případ, že se o tuto práci bude zajímat i čtenář z jiného než telekomunikačního oboru, se ale autor rozhodl tuto část do své práce zařadit alespoň jako součást příloh.

Druhá kapitola bude zaměřena na teoretické rozvedení problematiky otevřených řešení pro IP telefonii. Hlavním cílem této části bude seznámení se s otevřenými řešeními SIP serverů založených na protokolu SIP, popsání jejich možností a typického využití.

V třetí kapitole pak bude rozebrána problematika generování provozu na protokolu SIP a jeho doplnění o média (hlas). Hlavním tématem této kapitoly bude otevřený program SIPp a seznámení se s jeho funkcemi a vlastnostmi.

Čtvrtá část bude zaměřena na seznámení se se současným stavem problematiky testování SIP infrastruktury a výchozími body pro praktické řešení zadání diplomové práce. Touto kapitolou bude ukončena teoretická část této práce.

Pátá kapitola pak bude úvodem do praktického řešení problematiky výkonnostního testování infrastruktury založené na protokolu SIP. Bude pojednávat o obecném přístupu k problematice, o určení nutných podmínek pro testování, o základní terminologii pro testování a o významu jednotlivých termínů při interpretování výsledků testování.

Šestá část bude ve znamení vlastní realizace testovací. Budou zde rozebrány jednotlivé kroky nutné ke zprovoznění jak ústředny, tak testovacího softwaru SIPp. Rozebráno bude i samotné propojení jednotlivých prvků v síti a způsoby jejich vzájemného synchronizování a vzdáleného ovládání. Tato kapitola bude obsahovat i příklady částí skriptů a konfiguračních souborů, jejichž kompletní verze bude uvedena v přílohách na DVD.

Vlastní realizace testů, spouštěcí skripty a příkazy budou součástí sedmé kapitoly, ve které se čtenář také dozví o rozdílech v testovací topologii a v příkazech pro testování B2BUA a SIP Proxy.

Osmá kapitola uvede čtenáře do problematiky vyhodnocování nasbíraných výsledků. Bude ukázáno, které parametry jsou pro určení výkonu a limitů ústředny důležité a jak je číst. Tato část bude obsahovat příklady grafů.

V závěru pak bude tato práce zasazena do rámce dalšího vývoje a výzkumu. Bude zde nastíněno, jak bude autor pokračovat ve výzkumu dané oblasti a jaké jsou teoretické a praktické cíle, kterých chce autor dosáhnout.

2 OTEVŘENÁ ŘEŠENÍ PRO IP TELEFONII

Pod pojmem otevřené řešení se rozumí tzv. open sourcová realizace v tomto případě telefonní ústředny na bázi přepojování paketů. To znamená, že se v této kapitole budeme zabývat programy, které jsou šířeny pod licencí GNU GPL (General Public Licence), nebo některým z jejích derivátů. Hlavní výhodou takovýchto produktů je kromě ceny (neboť jsou zcela zdarma) zejména volně přístupný kód, což umožňuje prakticky komukoliv zapojit se do vývojových aktivit, čímž vzniká obrovská komunita spolupracujících vývojářů. Placené jsou až pokročilé služby – konfigurace, správa, propojení s klasickou telefonní sítí, apod. Otevřená softwarová řešení jsou obvykle určena pro některý z otevřených operačních systémů. Existují sice i portované verze, které fungují například i pod systémem Windows od firmy Microsoft, avšak nejvyššího výkonu a stability dosáhneme, pokud daný program implementujeme na počítači s operačním systémem Unix, Linux, popř. BSD.

Softwarových řešení, která z obyčejného počítače vytvoří softswitch na bázi přepojování paketů, existuje několik. Ovšem co do oblíbenosti vyčnívají zejména dvě řešení – Asterisk a Opensips (popř. Kamailio viz dále). Oba jmenované zástupce SIP serverů si nyní přiblížíme.

2.1 Asterisk

Asterisk je jedním z nejoblíbenějších řešení v oblasti telekomunikací přenášených IP protokolem. Asterisk je program, který je schopen z běžného počítače vytvořit telefonní ústřednu s poměrně rozsáhlými možnostmi, přičemž jeho výkonová náročnost není nikterak vysoká. Na low-endovém hardwaru (např. Athlon 64 X2) jsme schopni Asteriskem odbavit řádově desítky až stovky současných hovorů, přičemž existují i implementace, které jsou schopny integrovat Asterisk do hardwaru normálně používaného pro routery – příkladem je třeba projekt openWRT.

Asterisk je vyvíjen od roku 1999 a za otce celého projektu je považován Mark Spencer, který je zakladatelem firmy Digium, pod jejíž křídla Asterisk patří. Asterisk je součástí mnoha linuxových distribucí, jmenovitě například Debian, Ubuntu, Red Hat, Fedora a mnoho dalších. Krom tohoto je Asterisk dostupný i jako samostatné softwarové řešení, které v sobě integruje operační systém, webové rozhraní pro snadnou konfiguraci a další nástavby. Příkladem takového řešení je produkt firmy Fonality, který nese název Trixbox, popřípadě jeho konkurent AsteriskNow od firmy Digium. Oba systémy jsou postaveny na minimalistické úpravě linuxové distribuce CentOS a nepotřebují tedy pro své fungování žádné další softwarové doplňky.

Mezi klíčové vlastnosti Asterisku patří jeho komplexnost. Asterisk podporuje nejen paketově založené sítě (SIP, H.323, IAX), ale i digitální (ISDN) a analogovou (PSTN) telefonii. Podpora digitální a analogové telefonie je umožněna pomocí speciálních přídavných karet, za které již ale případný zájemce musí zaplatit, stejně jako za propojení do jmenovaných telefonních sítí.

Z hlediska VoIP se na stránkách společnosti Digium dále dočteme, že Asterisk disponuje následujícími vlastnostmi [2]:

- Podpora rozličných VoIP protokolů,

- Směrování řízení příchozích hovorů,
- Směrování a generování odchozích hovorů,
- IVR (Interactive Voice Response),
- Funkce pro správu médií,
- CDR (Call detail recording) záznamy pro tarifikaci,
- Konverze protokolů,
- Konverze médií,
- Integrovaná databáze,
- Skriptovací jazyk pro tvorbu dialplánu (AEL).

Asterisk samozřejmě disponuje i dalšími důležitými vlastnostmi, ale z hlediska této práce je nastíněný rozsah možností více než dostačující. Pro tuto práci klíčová vlastnost je skryta pod poměrně nenápadným heslem „konverze médií“. Jak se čtenář v dalších kapitolách dozví, právě tato vlastnost a její efektivita je zkoumána z pohledu jejího vlivu na výkon Asterisku v prostředích, kde koncoví klienti využívají různé kodeky pro přenos hlasu. Mezi jinými, Asterisk ve své základní konfiguraci podporuje kodeky G.711 (μ-law i A-law), G.726 a GSM. Jmenované kodeky budou využity při již zmiňovaném měření výkonu Asterisku.

Šíře schopností Asterisku jej předurčuje pro nejrůznější možná využití. Lze se setkat s Asteriskem ve funkci pobočkové ústředny (PBX), nebo brány (gateway), další uplatnění pak Asterisk nachází v Call centrech.

2.2 Opensips – představitel platformy SER

Zatímco Asterisk vyniká svou komplexností, Opensips exceluje svou jednoduchostí. Pod jednoduchostí by si však čtenář neměl představovat snadnost konfigurace, která je naopak ve srovnání s Asteriskem složitější. Jednoduchost, jak je zmíněna, je zde ve smyslu opaku komplexnosti. Kde Asterisk implementuje řadu konfiguračních souborů, které každý řeší úzkou oblast činnosti Asterisku, tam Opensips přichází se dvěma konfiguračními soubory, přičemž jedním uživatel ovládá prakticky celou činnost programu.

Opensips je příkladem softwarové SIP Proxy, která, jak je vysvětleno v příloze P I, se stará pouze o signalizaci. Opensips vznikl jako odnož projektu OpenSER poté, co se vývojářský tým OpenSERu kvůli několika roztržkám rozpadl. Část původního týmu začala pracovat na projektu Opensips, část se vrhla do vytváření konkurenčního Kamailia, avšak vzhledem ke společným kořenům jsou si oba programy doposud velmi podobné. S novou verzí Kamailia to ale vypadá, že tato větev vývoje přeci jen začíná mít navrch a pro další aplikace bude Kamailio vhodnější. V průběhu řešení této práce byly ještě oba projekty rovnocenné, ale vzhledem k dostupnosti již rozsáhlých zkušeností s Opensips jsme se po diskuzi s vedoucím rozhodli zvolit pro ověření metodiky výkonnostních testů Opensips.

Opensips na rozdíl od Asterisku není jedním kompaktním programem, nýbrž souborem několika knihoven, které jsou vzájemně propojeny a volány prostřednictvím konfiguračního souboru. Tento konfigurační soubor (opensips.cfg) má pak formu programu, typické jsou programátorské funkce jako „if“, „case“, apod. Odsud pramení i problematičnost konfigurace

Opensipsu, neboť uživatel musí strávit poměrně hodně času učením se syntaxe a proměnných, které jsou v konfiguračním skriptu použity.

Opensips ve svém základu rozhodně neoplývá tolika vlastnostmi a funkcemi jako Asterisk, avšak další funkcionalitu je možné získat nahráním a kompilací dalších modulů. Takže Opensips jako SIP Proxy se může poměrně snadno stát i tzv. Media Proxy, pokud k němu doinstalujeme stejnojmenný modul. Uživatelská náročnost konfigurace se tímto úkonem ještě zvýší a hlavně rozšiřující moduly existují v mnoha verzích, přičemž každá z nich je napsaná pro jiný účel, což činí z konfigurace, ale i otestování a srovnání výkonu Opensipsu nesmírně náročný a komplikovaný úkon. Dále pak Opensips nepodporuje tak velké množství protokolů, soustředí se výlučně na SIP a podpory dalších protokolů dosáhneme až doinstalováním dalších modulů, popř. kombinací Opensipsu s Asteriskem.

Zdálo by se tedy, že vše nahrává Asterisku. Proč se tedy Opensips (popř. jeho obdoba ve formě Kamailia) ujal mezi odbornou veřejností a nebyl vytlačen Asteriskem? Důvod je jednoduchý – výkon a flexibilita. Flexibilitu umožňuje forma konfiguračního skriptu. Koncový uživatel si vlastně upravuje program dle svých potřeb a není omezen jen nastavením některých hodnot v konfiguračních souborech. Výkon pak vyplývá jednak z principu fungování SIP proxy, neboť absence médií v datovém toku procházejícím ústřednou výrazně snižuje hardwarové nároky každého hovoru, a dále pak ze způsobu, jakým Opensips funguje. Opensips je po spuštění převeden do formy binárního souboru, v němž je obsažena veškerá funkcionalita, proto se často stává, že výkon Opensipsu je omezován výkonem externích aplikací, jako například databází, apod. Pro srovnání – u Asterisku jsme si řekli, že i na low-end hardwaru zvládne odbavit až stovky současných hovorů, Opensips naproti tomu na stejném hardwaru odbaví řádově mezi deseti a dvaceti tisíci současných hovorů. V testech společnosti Transnexus pak Asterisk zvládl až tisíc současných hovorů, SER naproti tomu dokázal odbavit 200 hovorů za sekundu (současné hovory není možné vypočítat, neboť Transnexus využil proměnlivé pauzy).

3 GENEROVÁNÍ PROVOZU NA PROTOKOLU SIP

V oblasti výkonnostního testování infrastruktury založené na protokolu SIP je klíčové mít prostředky pro generování dostatečného množství SIP zpráv. Ve většině případů si ale se samostatnými zprávami nevystačíme, neboť je potřeba otestovat nejen schopnost přijmout a vyhodnotit konkrétní zprávu, ale i schopnost validace návaznosti zpráv a jejich položek ve stavovém stroji. Typickým příkladem je třeba autentizace, kdy jsou generátoru zasílány informace nutné pro úspěšné vytvoření kódu obsahujícího přihlašovací údaje. Proto musí generátor umožňovat jak odesílání předem nadefinovaných zpráv, tak i vyhodnocovat údaje, které přicházejí ze SIP serveru.

Dále je potřeba, aby generátor byl schopen pracovat s médii, tedy s vlastním hlasovým obsahem, která jsou uvnitř hovoru přenášena. Tato nutnost vyplývá zejména z toho, že je potřeba mít k dispozici nástroj nejen pro testování SIP proxy, ale také B2BUA, kde média procházející skrze ústřednu ji vytěžují více než samotná SIP signalizace.

Obě vlastnosti splňuje program SIPp, který je open sourcovým generátorem provozu na protokolu SIP. SIPp je nyní dostupné pro všechny operační systémy vycházející z unixové architektury, ale i pro Windows, pod kterými ale nedosahuje optimálního výkonu. Jedná se o velikostně malý program, který ale nabízí řadu možností pro generování SIP zpráv

První a nejdůležitější možností z hlediska testování SIP ústředen je jednoznačně generování signalizačních zpráv protokolu SIP. SIPp generuje a odesílá zprávy na základě předem připravených scénářů ve formátu *xml* souboru. Tento soubor má klasickou strukturu běžnou pro jazyk XML a jednotlivé tagy (značky) mají logická pojmenování – např. *send* pro zprávy, které mají být odeslány, a *recv* pro zprávy, které SIPp očekává. Při psaní tohoto xml souboru je možné využít korektor xml syntaxe dostupný na stránkách SIPp. Jedná se o soubor s příponou DTD, který umožňuje textovému editoru, ve kterém uživatel píše xml soubor, kontrolovat xml syntaxi. Jednotlivé SIP zprávy jsou pak v xml scénáři obsaženy v poli textu začínajícím frází *CDATA*. V tomto textovém poli uživatel píše přímo textové SIP zprávy, jak jsou definovány v RFC 3261. Doplněním těchto zpráv jsou pak pole ohraničená hranatými závorkami, která obsahují buďto informace, které uživatel zadá pomocí příkazové řádky, nebo interní informace programu SIPp (například číslo portu). Speciálním typem těchto klíčových polí jsou ta, která uvnitř hranatých závorek obsahují text „fieldN“, kde N je kladné celé číslo. Tato pole jsou odkazy na informace uložené v externím textovém souboru, který může obsahovat například přihlašovací údaje. Část xml souboru zobrazuje obr. 3.1. Kombinací jmenovaných prostředků umožňuje SIPp generovat SIP zprávy s proměnnými informacemi a simulovat tak reálný provoz.

Generování médií je pomocí programu SIPp poměrně složité. Program sám o sobě nemá schopnost otevřít například zvukové soubory zakódované PCM modulací (wav). Je to z toho důvodu, aby se snížila komplexnost programu, který by jinak musel obsahovat značné množství kodeků, kterými by načtené hudební soubory kódoval. Naproti tomu autoři zvolili poněkud jednodušší přístup, kdy si uživatel musí vygenerovat soubor s příponou *pcap*, který daná zvuková data ukládá již ve formě RTP paketů zakódovaných pomocí některého z kodeků. Tento typ souboru je možné získat sledováním provozu na síti například programem Wireshark. SIPp pak tento soubor načte a v něm jednotlivě zaznamenané RTP pakety odesílá. Na přijímací straně

pak SIPp umožňuje hovorová data přijmout a odeslat je ve stejné formě zpět, což je výhodné například pro zjišťování vlivu přenosové cesty na kvalitu hovoru.

Dalšími přednostmi SIPp je podpora TLS, IPv6 a schopnost zaznamenávat kompletní statistiky hovoru. Zde lze najít jednak základní charakteristiky hovoru jako např. délka hovoru, počet opakovaných odeslání nějaké zprávy, apod. Je možné si v rámci xml scénáře nadefinovat až pět časovačů a čítačů, které měří četnost dané zprávy za dobu běhu SIPp a dobu, která uplyne mezi zprávami, z nichž jedna je označena jako start časovače a druhá jako jeho konec.

SIPp je v současné době jediným nástrojem, který umožňuje skutečně robustní testovací scénáře pro testy SIP Proxy i B2BUA. Je logickou volbou každého, kdo chce výkonnost SIP serveru měřit, avšak jeho velkou nevýhodou je problematická optimalizace, kdy zejména klientský konec má enormní požadavky na výkon stroje, na kterém pracuje.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

  <scenario name="UAC with media for performance testing WITH TRANSCODING">

  <send start_rtd="1" start_rtd="3" counter="1" retrans="500">                <!-- REGISTER -->
  <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: [field0] <sip:[field0]@[remote_ip]:[local_port]>;tag=[call_number]
      To: [field0] <sip:[field0]@[remote_ip]:[local_port]>
      Call-ID: reg///[call_id]
      CSeq: 1 REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port]
      Content-Length: 0
      Expires: 90

  ]]>
  </send>

  <recv response="100">                                                      <!-- 100 Trying -->
  </recv>

  <recv response="401" auth="true" crlf="true">                              <!-- 401 Unauthorized -->
  </recv>

  <send retrans="500">                                                       <!-- REGISTER w authentication -->
  <![CDATA[

      REGISTER sip:[remote_ip] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: [field0] <sip:[field0]@[remote_ip]:[local_port]>;tag=[call_number]
```

Obr. 3.1: Část XML scénáře pro SIPp se zvýrazněnou syntaxí

4 SOUČASNÝ STAV VÝKONNOSTNÍHO TESTOVÁNÍ

Protokol SIP již je používán v oblasti VoIP telefonie od roku 1999, od této doby si byl schopný vydobýt pozici hlavního signalizačního protokolu. V architekturách sítí nové generace je se SIPem počítáno jako s jediným protokolem pro VoIP. Odsud je patrné, že se četnost využití řešení založených právě na protokolu SIP bude v budoucnu určitě zvyšovat, přesto v této oblasti neexistují jednotné a standardizované prostředky pro evaluaci a benchmarking výkonu SIP serveru. V současnosti, pokud si z nějakého důvodu potřebujeme ověřit výkonnost své ústředny založené na protokolu SIP, můžete sáhnout po některém z proprietárních řešení, například po zařízení Ixload, nebo se můžeme pokusit vyrobit si prostřednictvím otevřených programů vlastní softwarový tester, pak ale vyvstává problém s definicí veličin, které budeme měřit, a způsobů, jakými tyto veličiny budete měřit. V oblasti definování parametrů pro testování si ale lze pomoci několika rozpracovanými drafty mezinárodní standardizační organizace IETF. Nyní si rozebereme obě zmíněné možnosti, abychom si vytvořili ucelený obraz o současném stavu výkonnostního testování SIP infrastruktury.

4.1 IXIA Ixload

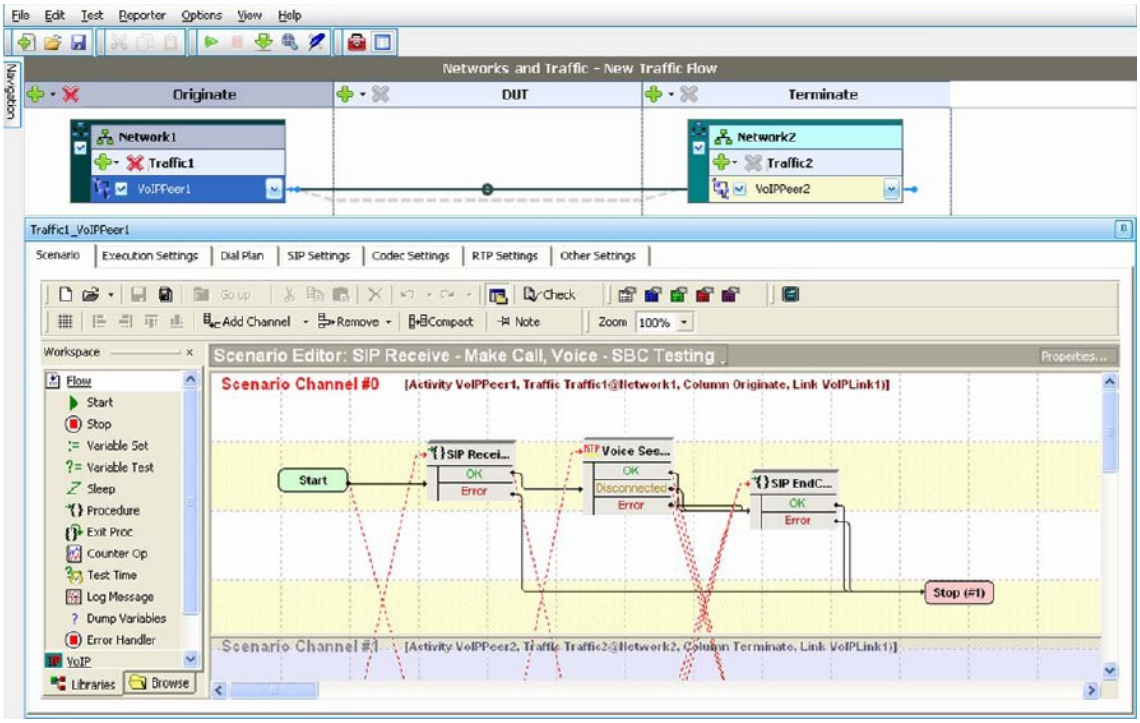
V úvodu této kapitoly bylo řečeno, že Ixload je jedním z možných řešení pro testování SIP infrastruktury. Ve skutečnosti ale zvládá mnohem více než jen toto. Firma Ixia, která Ixload vyvíjí a vyrábí, patří ke špičce v oblasti testování konvergovaných sítí na bázi IP protokolu, což dokazuje i hodnocení Frost and Sullivan, které za rok 2009 pasovalo Ixii do pozice absolutního leadera v oblasti výkonnostního testování [20]. Z tohoto důvodu jsou její produkty schopny obsáhnout široké spektrum scénářů a protokolů. Nejinak je tomu i u zařízení Ixload. Jedná se zařízení, které v sobě integruje pomocí modulárních prvků řadu funkcí – od nepostradatelné síťové konektivity, přes bloky umožňující generování zpráv jednotlivých protokolů, až po signálové procesory pro vyhodnocení kvality hovoru. Odsud plyne, že se jedná o poměrně komplexní řešení.

Uživatel při koupi obdrží zařízení o velikosti zhruba 3U serveru, které v sobě nese základní prvky pro propojení jednotlivých modulů, které si uživatel musí koupit zvlášť. Tyto moduly se liší v závislosti na měřených parametrech a veličinách. Pro testování SIP infrastruktury je třeba mít modul, který podporuje jednak SIP a jednak obsahuje prvky pro vyhodnocení kvality hovoru jak z pohledu SIPu, tak z pohledu kvality hlasu.

Ovládání je realizováno prostřednictvím grafického rozhraní, kde si uživatel může jednoduchým způsobem doslova „naklikat“ požadovaný scénář i měřené parametry, obr. 4.1 ukazuje náhled, jak může daný testovací scénář vypadat. Tato možnost je jistě velkou výhodou, neboť odpadá nutnost učit se složitou syntaxi příkazů apod.

Další výhodou je automatizace generování výstupů z měření. Po provedení měření software nasbíraná data sám vyhodnotí, utvoří tabulky a grafy a následně je vyexportuje do dokumentu ve formátu PDF. Uživatel má tak okamžitě k dispozici čitelnou podobu výsledků a nemusí sám řešit problém, jakým způsobem naměřená data dále zpracovat, čímž ušetří nemalé množství času.

Ve výše uvedených odstavcích jsme si ukázali mnohé z výhod nejen Ixload, ale proprietárních řešení obecně. Ve zkratce by se dalo říci, že nabízejí uživateli množství funkcí, které uživatel ovládá pomocí propracovaného a uživatelsky příjemného rozhraní. Tato řešení mají ale i nevýhody. Přejdeme-li pro mnohé klíčovou vlastnost, kterou je vysoká cena, pak z hlediska vědeckého a technického narazíme u srovnání výsledků naměřených pomocí zařízení různých výrobců. Neexistuje totiž standard, který by popisoval unifikovanou metodu měření, takže si každý výrobce na základě vlastního uvážení vytvoří svou metriku i metodiku, přičemž srovnatelnost těchto řešení různých výrobců je povětšinou malá.



Obr. 4.1: Příklad GUI přístroje Ixload[3]

Jinými slovy, v rámci jedné firmy nebo instituce je možné využít některé z proprietárních řešení pro určení výkonu několika provozovaných jednotlivých ústředen, nebo spojovací infrastruktury jako celku a pak tyto výsledky mezi sebou porovnat, avšak srovnání mezi různými subjekty, které vsadily na různá řešení, nedosáhneme, tudíž nelze sestavit nějaký žebříček hardwaru, či softwaru z hlediska jeho vhodnosti pro dané použití, nebo specifikovat parametry, které by o této vhodnosti vypovídaly.

Neexistence standardů v oblasti výkonnostního testování SIP infrastruktury je velký problém, avšak pomalu se v rámci IETF zpracovávají drafty, které se touto tematikou zabývají, je proto velmi pravděpodobné, že jakmile tyto drafty přejdou v plnohodnotné standardy, následně se i výrobci přizpůsobí a implementují parametry, které tyto standardy budou definovat.

4.2 Drafty IETF

Několik skupin se v rámci IETF zabývá problematikou SIPu a výkonnostního testování na něm založené infrastruktury. Zásadní význam pro tuto práci měly zejména tyto tři:

1. Methodology for Benchmarking SIP Networking Devices [4],
2. Terminology for Benchmarking Session Initiation Protocol (SIP) Networking Devices [5],
3. SIP End-to-End Performance Metrics [6].

Všechny tři drafty rozebírají podobnou problematiku a nyní si je rozebereme podrobněji a zasadíme je do kontextu s touto prací.

4.2.1 Methodology (Terminology) for Benchmarking SIP Networking Devices

Tento draft se kromě klasických definic významů jednotlivých termínů věnuje stanovení a popisu metod pro testování zařízení fungujících na protokolu SIP. Zabývá se jak testováním samostatných zařízení (DUT – Device under Test), tak i komplexních systémů (SUT – Systém under Test), přičemž zařízením se myslí zejména SIP Servery a systémem pak SIP Server v kombinaci s firewallem, popř. NATem. Metody definované v tomto draftu nejsou ovlivněny použitým transportním protokolem, což znamená, že mohou být aplikovány se všemi třemi variantami transportních protokolů, které SIP využívá, tedy UDP, TCP i TLS.

V další části tohoto draftu je čtenář seznámen se základní terminologií, jako je například množství generovaných relací za sekundu (Attempted Sessions per Second), nebo definice nulové a nekonečné relace. Za terminologií následuje vzorová zpráva z měření, která má za cíl zjednodušit následné porovnávání výsledků z různých měření, avšak definované možnosti nejsou zcela obecné a nemohou být použity ve všech případech. Dále je v tomto draftu popisováno měření několika veličin, které ale nejsou definovány uvnitř tohoto draftu. Doplněním tohoto draftu je totiž draft (2) „*Terminology for Benchmarking Session Initiation Protocol (SIP) Networking Devices* [5]". Tento draft definuje veškeré veličiny zmiňované v draftu (1) a mnoho dalších. V definicích najdeme jak základní popis veličin, tak i přesně vyjmenované SIP zprávy, které se k dané veličině vztahují. Vzájemně se tedy oba drafty doplňují a je vhodné k nim přistupovat jako k jednomu celku.

V poslední části se draft (1) zabývá definicí metod, jak měřit některé z definovaných parametrů. Jedná se o návod, který čtenáře vede doslova krok po kroku s cílem dosáhnout konkrétní měřené hodnoty. Zde je pak největší slabina tohoto draftu, neboť nastíněná metoda měření v některých případech nevede ke konvergenci ke skutečné hodnotě. Tento problém si ukážeme na konkrétním příkladu. Nejprve si ale projděme jednotlivé kroky měření tak, jak jsou definovány v draftu, pro příklad postačí nejjednodušší případ pro měření maxima počtu úspěšných relací (Session Establishment Rate; relaci je možno chápat jako termín zastupující množinu měřených veličin, např. proces registrace, výstavby spojení, atd.), jak je uveden v části 6.1 [4].

1. *Nakonfigurujte testované zařízení podle schématu v obrázku č. 1.*
2. *Nakonfigurujte tester, aby generoval 100 nových relací za sekundu při maximálním množství relací rovném 100 000. V generovaných relacích nesmí být přenášena média.*
3. *Začněte testovat zařízení (DUT) generováním nových relací.*
4. *Na testeru měřte neúspěšné pokusy o vytvoření nové relace a celkové množství úspěšných relací.*

5. *Pokud zjistíte, že došlo k neúspěšnému pokusu o vytvoření relace, snižte množství generovaných relací o 50%.*
6. *Pokud všechny relace byly úspěšně vytvořeny, zvyšte množství generovaných relací o 50%.*
7. *Opakujte kroky 3 až 6 dokud nedosáhnete výsledné hodnoty maximálního množství úspěšně vytvořených relací.*

Předpokládáme, že veškeré nastavované hodnoty budou celočíselné, proto zde vyvstává problém se zaokrouhlováním, který je možno řešit buďto využitím pouze celočíselné části výsledku dělení (zaokrouhlování dolů vždy), nebo zapisováním a sčítáním zbytků, přičemž se využije klasických pravidel pro zaokrouhlování, avšak ani tato úprava nemusí vést ke korektní konvergenci. Nedokonalost tohoto postupu ukazuje tab. 4.1.

Abychom se vyhnuli problémům se zbytkem při dělení a jeho zohlednění pro nastavovanou a cílovou hodnotu, je vhodné, aby zvolená škála neobsahovala čísla o základu deset. Mnohem vhodnější se ukazuje využití mocnin o základu dva, neboť při násobení i dělení dvěma vždy obdržíme celočíselný výsledek. Dále je nutné dosáhnout rychlé konvergence i pro mnohem vyšší cílové hodnoty než je základní hodnota nastavovaná na testeru v kroku číslo 1, proto je vhodné násobit o více než jen polovinu. Opět se tedy vyplatí násobit dvěma. Navrhovaný postup lze pak shrnout do těchto kroků:

1. Nakonfigurujte testované zařízení podle schématu v obrázku č. 1.
2. Nakonfigurujte tester, aby generoval 1 novou relaci za sekundu při maximálním množství relací rovném 100 000. V generovaných relacích nesmí být přenášena média.
3. Začněte testovat zařízení (DUT) generováním nových relací.
4. Na testeru měřte neúspěšné pokusy o vytvoření nové relace a celkové množství úspěšných relací.
5. Pokud nezjistíte neúspěšný pokus o vytvoření relace, zdvojnásobte množství generovaných relací.
6. Krok 5 opakujte, dokud nezjistíte neúspěšný pokus o vytvoření relace. Jakmile zjistíte první neúspěšný pokus o vytvoření relace, pokračujte ke kroku 7.
7. Pokud zjistíte, že došlo k neúspěšnému pokusu o vytvoření relace, snižte množství generovaných relací o 50 % rozdílu mezi poslední a předposlední nastavenou hodnotou množství generovaných relací.
8. Pokud všechny relace byly úspěšně vytvořeny, zvyšte množství generovaných relací o 50 % rozdílu mezi poslední a předposlední nastavenou hodnotou množství generovaných relací.
9. Opakujte kroky 7 a 8 dokud nedosáhnete výsledné hodnoty maximálního množství úspěšně vytvořených relací.

Cílovou hodnotu rozeznáme podle toho, že v posledním kroku přičteme, nebo odečteme jedničku, což vyplývá z vlastností celočíselných násobků čísla 2. V kroku 2 je z důvodu definování co nejobecnější metody pro určení hledané hodnoty řečeno, aby na začátku měření byla nastavena hodnota jedné generované relace za sekundu, což může vést ke zdlouhavému hledání měřené hodnoty, pokud je tato hodnota mnohonásobně vyšší. Z tohoto důvodu je možné

počáteční hodnotu nastavit na základě kvalifikovaného odhadu na tu z mocnin čísla dvě, o které si myslíme, že je nižší než hledaná hodnota. V případě, že detekujeme chybu hned v prvním kroku, pak víme, že námi odhadnutá hodnota byla špatná.

Tab. 4.1 ukazuje názorně rozdíl mezi oběma představenými metodami. Předpokládáme zařízení, které je schopné zvládnout 65 nových relací za sekundu a zatímco pomocí metody uvedené v draftu při využití celočíselného dělení (zaokrouhlovaní dolů vždy) čísla 65 nikdy nedosáhneme, protože dojde k zacyklení v kroku číslo 21, pomocí nově definované metody dosáhneme čísla 65 v kroku číslo 14. Pokud bychom předpokládali počáteční hodnotu jako 32, pak bychom počet kroků k určení měřené hodnoty snížili na devět. Písmenka S a F vyjadřují, zda v daném kroku došlo k neúspěšnému pokusu o vytvoření relace. Jedná se o první písmena anglických slov Successful (úspěšný krok, nedošlo k neúspěšnému pokusu o vytvoření relace) a Failed (neúspěšný krok, objevil se neúspěšný pokus o vytvoření relace).

Metoda navrhovaná draftem			
#	**Rel./s**	**S/F**	**Inkr./ Dekr.**
1	100	F	-50
2	50	S	+25
3	75	F	-38
4	37	S	+18
5	55	S	+27
6	82	F	-41
7	41	S	+20
8	61	S	+30
9	91	F	-46
10	45	S	+22
11	67	F	-34
12	33	S	+16
13	49	S	+24
14	73	F	-37
15	36	S	+18
16	54	S	+27
17	81	F	-41
18	40	S	+20
19	60	S	+30
20	90	F	-45
21	45	S	+22

Nově navrhovaná metoda			
#	**Rel./s**	**S/F**	**Inkr./ Dekr.**
1	1	S	+1
2	2	S	+2
3	4	S	+4
4	8	S	+8
5	16	S	+16
6	32	S	+32
7	64	S	+64
8	128	F	-32
9	96	F	-16
10	80	F	-8
11	72	F	-4
12	68	F	-2
13	66	F	-1
14	65	S	-

Tab. 4.1: Příklad hledání měřené hodnoty pomocí dvou porovnávaných metod

Rozebíraný draft, jak je patrné, obsahuje celou řadu nedokonalostí a je použitelný jen pro dílčí měření, proto pro tuto práci prakticky nebyl využit. Přesto je tento dokument důležitý z hlediska nezkušeného uživatele, který se pokouší o vytvoření testovacího scénáře, neboť je výborným příkladem, jak uvažovat nad metodologií měření a nad zapisováním výsledků tak,

aby byly porovnatelné s dalšími měřeními. Zároveň poskytuje inspiraci pro základní úvahy o testování zařízení založených nejen na SIPu, neboť definuje základní měřený parametr – množství generovaných relací za sekundu.

4.2.2 SIP End-to-End Performance Metrics

Z pohledu této práce byl tento draft mnohem důležitější než dva předchozí. Jedná se o poměrně propracovaný dokument, který je vydán už ve své čtvrté verzi. V prvních části se opět dozvídáme o termínech, které se v tomto draftu vyskytují. Tato část je podobná předchozím dvěma draftům, avšak je rozšířena o některé termíny z RFC 3261.

Následuje část, která definuje časové intervaly, které jsou pro SIP významné. Dozvídáme se i o časech označených jako T1 a T4, přičemž T1 je definován jako doba vyslání prvního bitu SIP žádosti a T4 je doba přijetí posledního bitu příslušné SIP odpovědi.

Za definicí termínů a časů následuje samotný popis výkonnostní metriky, kde se dovídáme o jednotlivých veličinách, které lze měřit. Mezi jinými jsou zde definovány *„Registration Request Delay (RRD)“*, *„Ineffective Register Attempts (IRA)“* a *„Session Request Delay (SRD)“*, kterým se, vzhledem k tomu, že s nimi bude dále v praktické části této práce pracováno, nyní budeme věnovat více.

Podle popisovaného draftu je RRD definováno jako interval mezi finální odpovědí a žádostí Register. Jedná se o interval, který je měřen na straně UAC a to pouze pro úspěšně vyřízené žádosti Register, neboť pro neúspěšné žádosti je definována samostatná veličina. S využitím dříve popsaných časů T1 a T4 můžeme říci, že T1 je doba vyslání prvního bitu žádosti Register a T4 je doba přijetí finální odpovědi 200 OK na tuto žádost. Odsud vyplývá, že musí platit rovnice:

$$RRD = T4(200\ OK) - T1(Register) \quad [\text{ms; ms}] \tag{1}$$

RRD je v praktické části této práce využito v každém ze scénářů, neboť je vhodné pro sledování prodlužování odezev SIP serveru při zvyšování množství generovaných hovorů. Dále je důležité sledovat, kolik z odeslaných žádostí Register bylo úspěšně zpracováno, k tomuto účelu slouží veličina IRA. IRA je definovaná jako percentuálně vyjádřený podíl neúspěšných žádostí Register (R_F) a celkového množství těchto žádostí (R_C), platí tedy rovnice:

$$IRA = \frac{R_F}{R_C} \cdot 100 \quad [\%; \text{-}] \tag{2}$$

Takto tedy rozebíraný draft pracuje s metodou Register. Registrace je však testem SIP Registrar serveru a není možné ji využít pro evaluaci výkonu SIP proxy, proto je nutné využít i další z definovaných veličin – SRD.

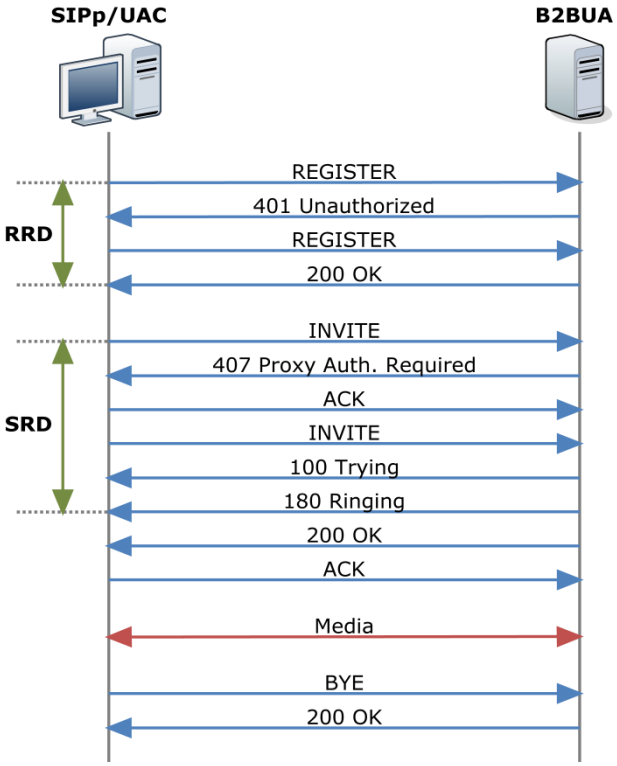
Stejně jako RRD je i SRD měřeno na UAC, avšak na rozdíl od RRD je měřeno pro oba případy – úspěšné i neúspěšné žádosti. Pro jednoduchost si rozebereme pouze úspěšné žádosti, neboť s těmi se setkáme ve většině případů. SRD je opět časovým intervalem mezi dvěma časy T1 a T4, přičemž T1 je definován jako čas vyslání prvního bitu žádosti Invite a T4 je pak čas přijetí posledního bitu prozatímní odpovědi jiné než 100 Trying. V běžném telefonním hovoru

se jedná o zprávu 180 Ringing, takže SRD pro úspěšně vyřízenou žádost Invite je definováno rovnicí:

$$SRD = T4(180\ Ringing) - T1(Invite) \quad [\text{ms; ms}] \tag{3}$$

Při měření obou hlavních veličin RRD i SRD se bere v úvahu pouze první vyslaná žádost a čas jejího vyslání, to znamená, že žádná z následně přeposlaných žádostí se stejným Call-ID není brána v úvahu, a to z toho důvodu, že při vyšších zatíženích nemusí být SIP server schopný odpovědět v definované době, ačkoliv je stále schopen úspěšně zpracovat žádost a správně ji směrovat.

RRD i SRD jako časové intervaly mezi souvisejícími SIP zprávami jsou názorně zobrazeny na obr. 4.2.



Obr. 4.2: RRD a SRD v kontextu příslušných SIP zpráv

Kromě dalších veličin je v draftu i vysvětleno několik aspektů měření, zejména vliv využití SIP Serveru v konfiguraci B2BUA, kdy se SIP Server chová jako oba konce spojení – UAC i UAS. Draft proto uvádí, že v některých případech, kdy je v rámci topologie nasazen B2BUA, je vhodné provádět měření na obou stranách spojení, avšak pro potřeby této práce si vystačíme s měřením pouze na konci, který zahajuje spojení.

Uvedený draft tedy vytváří rámec pro objektivní měření a srovnávání výkonů SIP Serverů, z pohledu dosud uvedených možností se jedná o nejkvalitnější a nejdostupnější řešení problematiky výkonnostního testování SIP infrastruktury. O propracovanosti a vhodnosti tohoto draftu svědčí i využití některých z draftových myšlenek a veličin v praktických měřeních prováděných firmou Transnexus (viz [7]). Tato firma provedla několik měření výkonnosti SIP serverů, přičemž využila open source řešení, o kterém již zde padla zmínka, SIPp, přičemž SIPp

bylo nakonfigurováno tak, aby měřilo, kromě jiného, právě časové intervaly mezi SIP zprávami, avšak zde pravděpodobná inspirace tímto draftem končí, neboť jsou měřeny intervaly mezi žádostí Invite a odpovědí 100 Trying, a následně pak až mezi 100 Trying a 180 Ringing. Ačkoliv se Transnexus zcela nedrží představeného draftu (v době měření existoval teprve ve své první verzi), přesto je řešení představené firmou Transnexus z pohledu metody měření velmi vhodné, neboť jej lze provést nezávisle na dostupném hardwaru a jeho konkrétní parametry mohou být upraveny. Z těchto důvodů se tato práce do značné míry inspirovala právě řešením, které firma Transnexus představila, a které je možné shlédnout v kompletní verzi jak pro B2BUA, tak pro SIP Proxy (viz [8], [9]).

Touto částí uzavíráme teoretický rozbor problematiky výkonnostního testování, SIPu a VoIP obecně. Předchozí kapitoly měly za cíl postupnými kroky seznámit čtenáře s informacemi nezbytnými pro pochopení následujících částí. V následujících kapitolách si vysvětlíme konkrétní řešení vycházející ze dříve zmíněných teoretických poznatků, provedeme instalaci a konfiguraci všech zařízení a programů, které budeme potřebovat, následně se zaměříme na samotné testování výkonnosti SIP serverů a vyhodnocení získaných dat. Kompletní soubor grafů pak čtenář nalezne v přílohách.

5 ZÁKLADNÍ MYŠLENKY A CÍLE VÝK. TESTOVÁNÍ

Z teoretických kapitol, které předcházely této, již víme, jakým směrem se budeme ubírat při naší snaze o vytvoření řešení pro testování výkonu SIP serverů. V rámci obecnosti se budeme snažit vycházet z draftu (3) rozebraného v kapitole číslo 4, pro udržení rozumné míry finanční náročnosti se zase omezíme na open source řešení SIP serverů i testovacích nástrojů, což nás přiblíží realitě firem, které mají k dispozici rozličný hardware a nemusí být schopny si pořídit některé z drahých proprietárních řešení. Jinými slovy víme, co budeme měřit, jak budeme měřit i čím budeme měřit. Přesto, ještě než začneme s vlastním nastavováním, je třeba se zamyslet nad několika aspekty samotného měření.

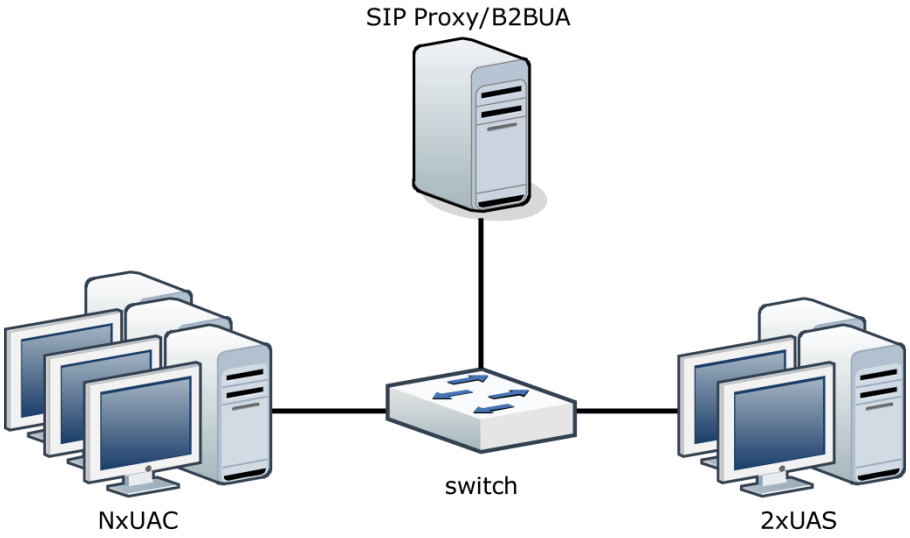
Především je vhodné seznámit se s možnostmi zařízení, která máme k dispozici. Jejich výkon a možnosti jsou jednoznačným limitujícím faktorem. Pro představu, nemůžeme chtít testovat high-end SIP server low-endovými prostředky, proto je nutné, aby testovaná i testovací zařízení byla minimálně na podobné technologické úrovni, přičemž je vhodné, aby testovací zařízení svým výkonem ta testovaná převyšovala. Je-li tato podmínka splněna, pak se nemusíme obávat, že v polovině testu zjistíme, že výsledky, které jsme měřením získali, jsou znehodnoceny chybami generovanými měřicími nástroji. Jak jsme si řekli v minulých kapitolách, budeme pro vlastní generování zátěže využívat program SIPp. Tento program je sice výborným pomocníkem pro měření, avšak není zcela vyladěn, co se optimalizace výkonu týče. Tento program je schopen využívat pouze jediné jádro procesoru a zároveň při vyšších zátěžích si řekne o značné množství paměti, což jsou omezení, se kterými je nutno počítat. Proto **je vhodné pro SIPp vyhradit několik samostatných počítačů**, nebo patřičným způsobem rozdělit výkon high-endového hardwaru mezi jednotlivé instance SIPp. V našem případě byly využity obě možnosti, neboť testy byl prováděny jak pomocí několika (konkrétně 6ti) fyzických PC, tak pomocí jednoho výkonného serveru, na kterém spolu koexistovalo několik virtuálních PC. Parametry SIPp, které je potřeba brát v úvahu si shrneme ještě v odrážkách:

- SIPp je čistě jednovláknová aplikace.
- SIPp není vhodné používat na jednom PC v několika instancích, neboť může docházet ke kolizím portů.
- SIPp, které funguje jako UAC, má několikanásobně vyšší požadavky na výkon počítače než SIPp, které funguje jako UAS.
- Je-li SIPp využito i ke generování médií, jeho požadavky dále rostou.
- Nehledě na výkon použitého počítače, v dále uvedených scénářích bylo SIPp schopno generovat řádově několik desítek hovorů za sekundu pro B2BUA a několik set hovorů za sekundu pro SIP proxy.

Vzhledem k omezením programu SIPp, které jsou uvedeny výše, bude celý systém schopen bezchybně testovat pouze ústředny se středním a nižším výkonem. V případě potřeby testování výkonnějších zařízení je vhodnější využít varianty s virtuálními počítači, neboť je toto řešení snáze škálovatelné.

Z pohledu měření pak vyvstává problém s opakovatelností měření, který je způsoben tím, že topologie sítě se v různých místech a firmách značně liší, proto je nutné vytvořit co nejjednodušší a nejsnáze opakovatelnou fyzickou topologii. Příkladem takové topologie je

zapojení na obr. 5.1, který zobrazuje fyzickou topologii při využití více fyzických počítačů. V případě využití virtualizace je fyzická topologie ještě jednodušší, avšak vždy musí být využit propojovací prvek (switch) tak, aby výsledky byly porovnatelné a nelišily se právě o zpoždění vytvořené tímto propojovacím prvkem.



Obr. 5.1: Topologie sítě využívané pro testování

Zásadní vliv mají i vlastnosti propojovacího prvku a síťových karet jednotlivých počítačů. V rámci omezeného měření (uvnitř firmy, domu) je jedno, jaká rychlost je použita, avšak pro porovnání výsledků v širší komunitě je třeba shodnout se na jediné používané rychlosti. V rámci této práce bylo využito 100 Mbit/s switche pro fyzickou topologii více počítačů a 1 Gbit/s switche pro řešení s využitím virtuálních PC. Různá zařízení byla použita vzhledem k dostupnosti a pro výsledky nemají degradující efekt, neboť každé zařízení bylo použito pro jiný případ testování (B2BUA, SIP proxy), kdy se ani generovaná zátěž nedá srovnávat. Přesto by se v dnešní době relativní dostupnosti 1 Gbit/s switchů měly používat právě tyto.

Z hlediska opakovatelnosti měření vyvstává ještě jeden problém. Oba testované SIP servery nabízejí velice široké možnosti vlastních úprav a přizpůsobení potřebám uživatele. Je možné si vytvářet složité dialplany a definovat vlastní funkce, které mají být aplikovány při nejrůznějších událostech. Takto široká modifikovatelnost z hlediska opakovatelnosti měření a srovnatelnosti výsledků škodí. Z tohoto důvodu jsem se v této práci zaměřil na testování obou ústředen v podstatě v nastavení, které je dodáváno přímo od výrobce a které jsem se snažil modifikovat jen minimálně. Tento postup ale není možné použít vždy, neboť základní nastavení se mnohdy velmi liší od provozního nastavení a tak by případnému uživateli výsledky mnoho neřekly. Přesto lze tyto výsledky využít právě pro srovnání jednotlivých ústředen z hlediska hardwarových a softwarových platforem.

Posledním problémem, nad kterým by se měl případný zájemce o provedení výkonnostních testů zamyslet, jsou měřené veličiny. Nyní nemám na mysli definici veličin, to za nás provedl draft (3), ale určení veličin, které budeme měřit. Není totiž nutné provádět měření všech v draftu definovaných veličin, v mnohých případech si vystačíme například s maximálním množstvím hovorů, které je schopna ústředna úspěšně odbavit. V rámci této

práce ale bude rozebráno měření všech hlavních parametrů, které se dají měřit. Pro zjednodušení si tyto parametry rozdělíme do dvou skupin:

- Statické parametry,
- Dynamické parametry.

Statickými parametry budou míněna veškerá maxima, jako maximum úspěšných registrací, maximum odbavených hovorů, apod. Jedná se o parametry, které jednoznačně určují výkon ústředny a které jsou tedy z hlediska jejího nasazení nejdůležitější. Naproti tomu dynamické parametry jsou ty veličiny, které se s proměnlivou zátěží mění. Jsou to vlastně veškeré parametry, které budeme měřit, avšak jejich využití není tak markantní jako u statických parametrů, neboť pomocí dynamických parametrů již budeme pouze „dolaďovat“ závěry z měření. Z uvedeného vyplývá, že statické parametry jsou vlastně limitními případy dynamických parametrů.

Nyní jsme prošli základními úvahami o měření a můžeme se tedy dát do samotné přípravy testovacího prostředí.

6 REALIZACE TESTOVACÍ PLATFORMY

Testovací topologie sestává z několika počítačů (viz obr. 5.1) a propojovacího switche, do jehož funkce zasahovat nemusíme, takže zbývá nastavení a příprava počítačů, přičemž na jednom z nich bude na pozadí fungovat SIP server a na zbylých pak SIPp. Počítač se SIP serverem je velmi důležitý, neboť jeho výkonnost bude klíčová pro naměřené výsledky, proto si nyní shrneme jeho technické parametry:

- CPU AMD Athlon 64 X2 5200+,
- RAM 4 GB DDR2,
- LAN 2x 1Gbit/s Ethernet,
- HDD 500 GB,
- OS Debian 5.01 (Lenny) 32-bit.

Počítače, které budou generovat zátěž pomocí programu SIPp, nejsou z hlediska svého výkonu tak významné z důvodů, které byly popsány v minulé kapitole, proto jejich specifikace nebudu uvádět. Důležité je pouze použitý operační systém, kterým bylo **Ubuntu 9.04 64-bit**. Stejný postup byl otestován i na následující verzi Ubuntu 9.10.

Veškeré úpravy, které si v následujících částech této kapitoly uvedeme, budou prováděny na čistých instalacích uvedených operačních systémů, takže uvedené postupy budou obsahovat všechny nezbytné kroky ke zprovoznění testovací topologie. Následující podkapitoly budou řazeny logicky tak, jak jsem postupoval při zprovozňování jednotlivých služeb.

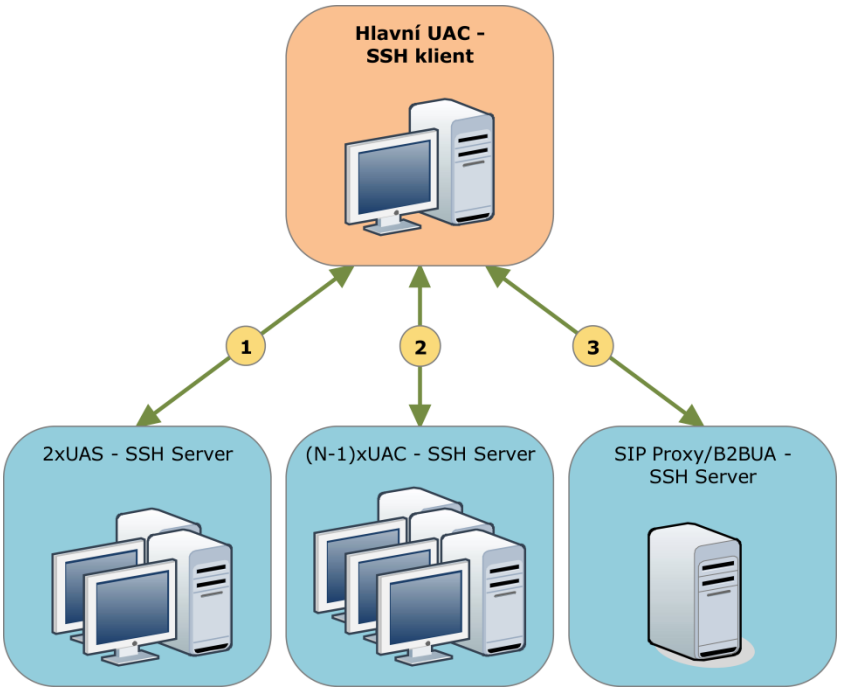
6.1 SSH

Prvním krokem ke zprovoznění testovací topologie je instalace a nastavení služby SSH. SSH je nástroj pro vzdálený přístup k počítači a umožňuje, aby si počítače mezi sebou vyměňovaly informace a příkazy. Doplňkem této služby je pak i možnost bezpečného kopírování, prostřednictvím příkazu *scp*, který budeme také hojně využívat zejména k přenosu výsledných dat.

SSH funguje v režimu server-klient, proto je potřeba si rozmyslet, které počítače budou v síti zastávat kterou úlohu. Formálně nejčistším řešením by bylo, kdyby v topologii existoval jeden počítač, který by se staral pouze o předávání příkazů ostatním počítačům a který by následně i shromažďoval data po skončení měření. Toto řešení je použito například i pro testy IMS Bench variantou programu SIPp [10]. Avšak toto řešení si žádá jeden samostatný počítač, který by pro testování jinak neměl žádnou úlohu, proto je vhodné využít některého z počítačů, na kterém pracuje program SIPp. Problematické ovlivňování měření na pozadí běžícím skriptem, který by pomocí SSH kontroloval celý průběh testu a způsoboval tak dodatečnou zátěž na jednom z měřicích počítačů, lze minimalizovat vhodnou skladbou tohoto skriptu, tzn. příkazy ostatním počítačům předávat před začátkem, nebo až po skončení samotného generování zátěže. Jediným příkazem, který takto nemůžeme odstranit z aktivní části skriptu je spuštění měřicího nástroje SAR (viz dále), toto však nepředstavuje významnější problém z hlediska zátěže.

Podle výše uvedeného se celá topologie tedy rozdělí na několik SSH serverů a jednoho SSH klienta, který bude koordinovat součinnost jednotlivých počítačů. Obr. 6.1 pak nastiňuje,

v jakém pořadí dochází ke komunikaci mezi servery a klientem v kontextu ústředny a počítačů se SIPp.



Obr. 6.1: Souslednost příkazů předávaných pomocí SSH

Instalace SSH klienta i SSH serveru je velmi jednoduchá, neboť se příslušný software nachází v repozitářích Debianu a tedy i Ubuntu, v obou systémech stačí tedy zadat:

```
# aptitude install openssh-server
```

resp.

```
# aptitude install openssh-client
```

Veškeré závislosti jsou pak vyřešeny balíčkovacím nástrojem *Aptitude* a není třeba se jimi zabývat.

6.1.1 Konfigurace SSH klienta

SSH klientem tedy bude jeden z počítačů generujících zátěž pomocí programu SIPp, v rámci dalšího zjednodušení budu ve shodě s obr. 6.1 používat termín „hlavní UAC“, neboť se bude jednat o počítač, který bude fungovat jako UAC a zároveň bude předávat příkazy ostatním počítačům.

Po instalaci balíčků openssh na všech počítačích, které spolu mají komunikovat, lze již provést spojení prostřednictvím příkazu:

```
# ssh uživatel@IPadresa_ssh_serveru
```

Po zadání tohoto příkazu však vždy budeme požádáni o zadání hesla pro přístup ke vzdálenému počítači, což je vzhledem k nutnosti automatizace měření nežádoucí, proto musíme oběma stranám komunikace poskytnout klíče, které ověří jejich totožnost a umožní SSH spojení bez zadávání hesel. Vygenerovat tyto klíče lze příkazem:

```
# ssh-keygen -t rsa
```

Po zadání tohoto příkazu nás systém vyzve k vložení fráze, pomocí které se daný klíč vytvoří, tuto hodnotu ale není třeba vyplňovat. Pro generování klíčů byla použita šifra RSA, lze ale použít i šifru DSA.

Vygenerované klíče najdeme v uživatelském adresáři. Vzhledem k tomu, že jsme příkaz zadávali jako root (hash na začátku řádku), tak klíče nalezneme v adresáři */root/.ssh/*. Jedná se o dva soubory *id_rsa* a *id_rsa.pub*. První z klíčů je soukromý klíč klienta, se kterým nebudeme nic provádět, druhý je pak veřejný klíč, který je třeba importovat na vzdálený počítač (server).

6.1.2 Konfigurace SSH serveru

Veřejný klíč, který jsme vygenerovali na klientovi v předchozím kroku, je třeba zkopírovat na server. Jedná se pouze o dočasný soubor, tak nezáleží, kam jej zkopírujeme. Pro příklad jsem zvolil umístění */tmp/id_rsa.pub*. Informace uvedené v tomto souboru je třeba umístit do souboru *authorized_keys2* v pracovní složce uživatele, pod jehož jménem se budeme přihlašovat, tedy */root/.ssh/*. Tato složka však na SSH serveru ještě nemusí existovat, proto ji vytvoříme.

```
# mkdir /root/.ssh
```

Dále je potřeba změnit práva k této složce, aby s ní majitel (root) mohl provádět všechny operace.

```
# chmod 700 /root/.ssh
```

A nyní do této složky importujeme identifikační informace, což provedeme následujícím příkazem.

```
# cat /tmp/id_rsa.pub >> /root/.ssh/authorized_keys2
```

A opět změníme práva.

```
# chmod 640 /root/.ssh/authorized_keys2
```

Následně můžeme soubor *id_rsa.pub* smazat, je ale výhodné si ho ponechat pro případ dalšího využití.

Výše zmíněnými kroky jsme umožnili uživateli root na lokálním počítači (klient) přihlásit se jako uživatel root na počítači vzdáleném (server), aniž by musel zadávat heslo. Pro ostatní uživatele zůstává povinnost zadat heslo při iniciaci SSH spojení, což je nutné mít v paměti. Posledním krokem ke zprovoznění SSH bez hesla je umožnění vzdálenému uživateli přihlášení jako root. Tato možnost je defaultně u obou použitých systémů povolena, přesto je dobré ověřit, že v souboru */etc/ssh/sshd* je následující řádek.

```
PermitRootLogin       yes
```

Jakmile máme toto ověřeno, pak stačí na klientovi inicializovat nové SSH spojení již uvedeným příkazem:

```
# ssh uživatel@IPadresa_ssh_serveru
```

Po zadání toho příkazu se nás systém zeptá, jestli chceme připojující se počítač přidat do známých hostů. Toto odsouhlasíme zadáním „yes“, načež jsme přihlášeni na SSH serveru. Všechna další sezení již budou započata bez nutnosti jakéhokoliv zásahu ze strany uživatele.

6.1.3 Problém: Pomalé připojení přes SSH

Ačkoliv jsme provedli konfiguraci dle výše uvedeného návodu a SSH nám skutečně funguje, může se stát, že inicializační informace jsou vyměňovány neúměrně dlouho (řádově i v jednotkách minut). Tento problém bývá způsoben využitím SSH na sítích se statickými IP adresami, kde není definován server DNS, neboť SSH server provádí hledání v databázích DNS serveru. Tomuto je možné zamezit tím, že na straně serveru přidáme do souboru */etc/ssh/sshd_config* následující řádek.

```
UseDNS          no
```

Po restartování služby SSH na serveru následujícím příkazem by již vše mělo fungovat bez problémů.

```
# /etc/init.d/sshd restart
```

6.2 SIPp

Zatímco instalace a zprovoznění SSH byly poměrně jednoduché, v případě SIPp bude třeba vyřešit několik problémů. Prvním je instalace. Ačkoliv je SIPp v repozitářích Ubuntu pod jménem *sip-tester*, nová a z hlediska optimalizace jediná použitelná verze SIPp, tedy verze 3.1, je dostupná až v repozitářích Ubuntu 9.10. Ubuntu 9.04 má k dispozici starší SIPp 2.0.1. Zájemcům o skutečné výkonnostní testování ale novější Ubuntu 9.10 stejně práci neušetří, neboť pro zvýšení výkonu SIPp tak, aby bylo rozumně použitelné, je třeba sáhnout do instalačních souborů a provést samostatnou kompilaci. Následující návod je tedy shodný pro obě verze operačního systému Ubuntu.

6.2.1 Stažení, úpravy a kompilace

Pro stažení i rozbalení lze využít grafického prostředí Ubuntu, proto zde nebudu rozepisovat příkazy (wget, tar). Ze stránek SIPp si tedy stáhneme poslední verzi tohoto programu. V době vzniku této práce byla nejnovější verzí SIPp verze 3.1. Z nabízených možností stáhneme zdrojové kódy pro kompilaci, tedy soubor *sipp.3.1.src.tar.gz*. Po rozbalení tohoto archivu zjistíme, že obsahuje složku *svn*, ve které jsou umístěny všechny zdrojové soubory. Tuto složku si uložíme tam, kde budeme chtít náš program, v našem případě je to složka */home/loki/svn*.

V terminálu se jako root přesuneme do této složky. Změnu kompilačních souborů budeme provádět v programu gedit, tudíž pro změnu příslušného souboru použijeme tento příkaz.

```
# gedit jméno_souboru
```

Modifikovat budeme celkem 3 soubory. Prvním z nich je soubor *call.cpp*. V tomto souboru chybí odkaz na hlavičku, která obsahuje maximální a minimální hodnoty pro systémem definované atributy. Tento odkaz přidáme doplněním řádku

```
# include <limits.h>
```

hned na začátek souboru. Hash zde nemá význam uživatele root, ale jedná se o povinný začátek řádku s definicí, nebo odkazem. Dále je třeba opravit chybu, která způsobuje špatné mapování portu médií při vyjednávání spojení. Tato chyba má za následek, že hovor se nepodaří spojit, ačkoliv nepřijde žádná chybová zpráva. Důvod, proč k této chybě dochází, najdeme na řádku 195 (pokud jsme přidáním předchozího řádku nezměnili číslování). Tento řádek vypadá následovně:

```
begin += strlen(pattern) -1;
```

Chybu způsobuje „-1“ na konci tohoto řádku, neboť programu říká, že si má vzít z například pětimístného čísla portu pouze čtyři číslice. Problém tedy odstraníme jednoduše smazáním této „-1“.

Stejný problém s absencí hlavičky jako soubor *call.cpp* má i soubor *scenario.cpp*, proto do tohoto řádku stejným způsobem doplníme uvedený řádek.

Poslední úpravou, kterou musíme na zdrojových souborech SIPp provést, je přepsání vnitřně definované hodnoty pro maximální počet otevřených souborů. Tento limit má SIPp defaultně nastaven na 1024, což je problém při vyšší zátěži, kdy je programem často překračován. Tento vnitřní limit je obdobou „file descriptorů“ v operačním systému a nachází se v souboru *sipp.hpp* přibližně kolem řádku 80. Jedná se o následující souvislý blok:

```
# ifndef __CYGWIN
# ifndef FD_SETSIZE
# define FD_SETSIZE 65000
# endif
# else
# ifndef __CYGWIN
# ifndef FD_SETSIZE
# define FD_SETSIZE 1024
# endif
# endif
```

Problém je způsoben číslem 1024, vzhledem k tomu, že se ale nachází ve složitějším bloku, kde není zcela jasné, které z definic jsou volány kdy, je bezpečnější celý blok zakomentovat a za něj vložit novou definici proměnné FD_SETSIZE. Blok pak bude vypadat následovně:

```
/*
# ifndef __CYGWIN
# ifndef FD_SETSIZE
# define FD_SETSIZE 65000
# endif
# else
# ifndef __CYGWIN
# ifndef FD_SETSIZE
# define FD_SETSIZE 1024
```

```
# endif
# endif
*/

# define FD_SETSIZE 65000
```

Těmito změnami zaručíme, že SIPp bude fungovat správně a zároveň že nám umožní dosáhnout vyšších zatížení, avšak před samotnou finální kompilací musíme vyřešit závislosti.

Ze stránek SIPp [11] si lze závislosti přečíst, avšak je třeba mít na paměti, že, kde je to možné, musíme instalovat balíčky v tzv. vývojářských verzích, neboť je nepotřebujeme pro již funkční program, ale pro zkompilování zdrojových kódů do funkčního programu. Instalaci balíčků, na kterých SIPp závisí, provedeme následujícími příkazy:

```
# aptitude install gcc
# aptitude install g++
# aptitude install libncurses-dev
# aptitude install openssl
# aptitude install libssl-dev
# aptitude install libpcap0.8-dev
# aptitude install libnet1-dev
# aptitude install libgsl0-dev
```

Těmito příkazy vyřešíme veškeré závislosti, které program SIPp má. Poslední příkaz instaluje knihovnu GSL (GNU Scientific Library), která není nezbytná pro chod SIPp, je-li ale nainstalována ještě před kompilací programu, pak umožní SIPp využít parametry matematických rozdělení v oddílu pro média, jinými slovy lze definovat proměnlivé délky hovorů.

Po provedení změn ve zdrojových souborech SIPp a nainstalování všech balíčků, na kterých SIPp závisí, můžeme přistoupit ke kompilaci. Na rozdíl od jiných programů, které je nutno napřed nakonfigurovat speciálním příkazem a až posléze kompilovat, SIPp je možno kompilovat přímo, což provedeme standardně pomocí příkazu *make* (neustále ale musíme být ve složce se zdrojovými soubory).

```
# make pcapplay_ossl
```

Parametry, které jsou u příkazu make uvedeny způsobí, že kompilovaná verze bude podporovat jak média (pcapplay), tak i TLS (ossl). První možnost je pro potřeby této práce nutnost, druhá je volitelná. V průběhu kompilace se mohou objevit zprávy „WARNING“, těm ale není třeba věnovat pozornost, problém nastává až se zprávou „ERROR“, avšak při dodržení uvedeného postupu by se tato zpráva neměla objevit.

Uvedeným postupem jsme vytvořili funkční program, který je umístěn ve složce */home/loki/svn/*, avšak ze systému není možné SIPp spustit jinde než z této složky, proto je vhodné do systému vložit informaci o příkazu *sipp*, který bude spouštět náš program. Toto provedeme vytvořením odkazu na binární soubor *sipp* (který vznikl kompilací) do složky */usr/bin/*. Toto provedeme následujícím příkazem.

```
# ln –s /home/loki/svn/sipp /usr/bin/sipp
```

Nyní již program bude fungovat standardně pouhým zadáním příkazu *sipp* do příkazové řádky. Je vhodné jej ale spouštět jako root, zejména pak na straně UAC.

6.2.2 Konfigurace a příprava SIPp pro fungování jako UAC

Zkompilovaný program lze již použít jako testovací nástroj, dokonce v sobě i obsahuje několik scénářů, které je možné využít pro jednoduché testování. My ale potřebujeme několik konkrétních souborů pro vyžití funkcionalit, které v SIPp v defaultní konfiguraci nejsou obsaženy. Nyní si ukážeme pouze tvar jednotlivých souborů a jak je vytvořit, neboť data uvnitř těchto souborů si uživatel může přizpůsobit dle potřeby. Výjimkou budou soubory scénářů, pro jejichž změny jsou potřeba značné znalosti SIPu i SIPp, takže je vhodné si je vysvětlit podrobněji. Veškeré jmenované soubory jsou k dispozici na DVD přiloženém k této práci.

6.2.2.1 Soubor hovorových dat

Nejprve budeme potřebovat soubor, který bude obsahovat média, jež mají být vysílána jako hovorová data. Tento soubor je možné vytvořit jednoduše. Počítač, na kterém máme nainstalovaný Wireshark, připojíme prostřednictvím hubu ke dvěma telefonům, mezi nimiž si zavoláme. Jako hovorová data je vhodné použít nějakou hudbu, která hraje bez přestávky. Tento hovor odchytíme programem Wireshark a aplikujeme na něj filtr:

```
rtp and ip.src==IPadresa_telefonu
```

Zachycená data je možné omezit na konkrétní délku v menu pro uložení souboru, pro potřeby této práce jsem vytvořil soubor s daty o délce 60 sekund. Hovor mezi telefony může probíhat na libovolném kodeku, avšak, jako zdrojový kodek byl použit kodek G.711 μ-law. Jméno souboru jsem zvolil jako *g711u.pcap* a umístil jsem jej do nově vytvořené složky */home/loki/sipp/*. Pro funkčnost je ale možné zvolit libovolné jméno i umístění.

6.2.2.2 Soubor informací o hovoru

Další nutností je podat SIPp informace o tom, jako jaké číslo má volat, jaká jména a hesla má použít pro autentizaci a kam má volat. Tyto informace lze vložit prostřednictvím příkazové řádky, avšak my potřebujeme velké množství těchto informací, což použití příkazové řádky vylučuje. Schůdnou možností je vygenerování tabulkového souboru ve formátu *csv*. K vytvoření tohoto souboru je možné použít libovolný textový editor, vhodnější je ale tabulkový procesor s možností volby oddělovačů polí (sloupců). Oddělovač, který SIPp akceptuje je středník.

První řádek tohoto souboru je slovo, které definuje, jakým způsobem má SIPp tento soubor číst. Možnosti jsou SEQUENTIAL, RANDOM a USER. První možnost říká SIPp, aby četlo soubor řádek po řádku a jakmile dojde na konec, aby začalo znovu od začátku. Možnost RANDOM pak znamená náhodné čtení a USER se použije pro uživatelem definovaný vzorec. My využijeme možnost SEQUENTIAL.

Další řádky obsahují informace, které chce uživatel do SIPp importovat, přičemž každý řádek je určen pro jeden hovor, tzn. první hovor si vezme informace z druhého řádku (první obsahuje slovo SEQUENTIAL), druhý hovor přečte třetí řádek, atd. Jednotlivé informace, které chceme vložit do jednoho hovoru, jsou pak umístěny do po sobě jdoucích sloupců. V prvním

sloupci bude umístěno číslo volajícího, tedy číslo, které je umístěno v poli *From* SIP hlavičky. Druhý sloupec bude obsahovat autentizací informace a třetí pak číslo volaného, tedy informaci, která se objeví v poli *To* SIP hlavičky. Po exportu z tabulkového procesoru do textového souboru *csv* pak tento soubor bude vypadat takto:

```
SEQUENTIAL;
10000; [authentication username=10000 password=uac];50000
10001; [authentication username=10000 password=uac];50001
…
```

Pro testování B2BUA není potřeba tak velkého množství klientů jako pro testování SIP proxy a to z toho důvodu, že množství současných hovorů bude řádově menší. Proto jsem v rámci své práce volil pro testování B2BUA 4 000 klientů a 4 000 serverů, celkem bylo tedy v databázi 8 000 účastníků, zatímco pro SIP Proxy jsem použil 72 000 účastníků. Konkrétní rozsahy čísel si uvedeme až v podkapitolách věnovaných právě testování B2BUA a SIP Proxy.

Takto vytvořený soubor *csv* uložíme opět do složky */home/loki/sipp/* a to pod jménem *csv.csv*.

6.2.2.3 Soubor scénáře

Již v představení SIPp jako nástroje pro generování provozu na protokolu SIP a RTP jsme si řekli, že ke své funkci využívá scénáře napsané v jazyce XML. Tyto scénáře definují souslednost a obsah zpráv, které SIPp odesílá a přijímá, dále je v nich možné provádět některé akce jako například prohledávání SIP hlaviček, přerušení hovoru, nebo odesílání RTP streamu.

Soubor XML začíná následujícími dvěma řádky, které definují kódování a DTD soubor pro kontrolu syntaxe.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">
```

Následuje řádek, který uvozuje celý blok XML dat, je možné do něj vložit i jméno scénáře jako parametr. Párová značka uvozující konec souboru začíná lomítkem. XML soubor je tedy ohraničen těmito dvěma řádky.

```
<scenario name="JMENO SCENARE">
</scenario>
```

Většina dalších značek (tagů) je v souboru párová a nyní si uvedeme jen příklad pro odeslání a přijetí SIP zprávy a použití parametrů s tímto spojených. Kompletní soubory pak lze stáhnout z přiloženého DVD.

Odeslání SIP zprávy si ukážeme na zprávě REGISTER. SIPp odešle tuto zprávu, pokud ve scénáři bude obsažen tento blok kódu:

```
<send start_rtd="1" start_rtd="3" counter="1"
retrans="500">
<![CDATA[
REGISTER sip:[remote_ip] SIP/2.0
Via:
```

```
SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
From: [field0]
<sip:[field0]@[remote_ip]:[local_port]>;tag=[call_number]
To: [field0] sip:[field0]@[remote_ip]:[local_port]
Call-ID: reg///[call_id]
CSeq: 1 REGISTER
Contact: sip:[field0]@[local_ip]:[local_port]
Content-Length: 0
Expires: 90
]]>
</send>
```

Opět je celý blok pro odeslání ohraničen párovými tagy <send> a </send>. Uvnitř tohoto tagu nalezneme parametry *start_rtd="1"* a *start_rtd="3"*, které spouštějí interní časovače v SIPp a to časovače 1 a 3. Parametr *counter="1"* pak inkrementuje hodnotu čítače číslo 1, na konci testu tedy v tomto čítači bude uložen počet odeslaných zpráv REGISTER. Parametr *retrans="500"* říká SIPp, aby zprávu přeposlalo v případě, že odpověď nepřijde do 500 ms.

Zbytek bloku je pak samotný text zprávy REGISTER a to ve formátu, jak je definována v RFC 3261. Řada parametrů je však přiřazována dynamicky, neboť je potřeba, aby SIPp odbavovalo množství hovorů s různými parametry. Jednotlivá pole, která jsou dynamicky přiřazována SIPp, jsou ohraničena hranatými závorkami a jejich významy jsou tyto:

- *remote_ip* – je pole obsahující IP adresu SIP serveru, hodnota tohoto pole se zadává v příkazové řádce (viz dále),
- *transport* – obsahuje informaci o protokolu transportní vrstvy (UDP, TCP, TLS),
- *local_ip* – toto pole je opět zadáváno přes příkazovou řádku a to prostřednictvím přepínače *„-i"* a jedná se o IP adresu počítače, na kterém je SIPp spuštěno,
- *local_port* – určuje číslo portu pro signalizaci SIP, standardně 5060,
- *branch* – branch vygenerované SIPp,
- *call_id* – Call-ID, opět generováno SIPp, část *„reg///"* před tímto polem je využita z důvodu odlišení Call-ID pro registraci a pro vlastní hovor, neboť se jedná o dva různé dialogy a v případě stejného Call-ID by hovor nebyl úspěšný,
- *fieldX* – odkazuje na X-tý sloupec v *csv* souboru.

Přijetí zprávy scénář ošetřuje tagy *<recv>* a *</recv>*, na straně UAC v nich nejsou obsaženy žádné textové informace. Přijetí zprávy *401 Unauthorized*, která je odpovědí na zaslanou zprávu REGISTER je ve scénáři popsáno takto:

```
<recv response="401" auth="true" crlf="true">
</recv>
```

Pole *response="401"* definuje zprávu, která má být přijata. Pole *auth="true"* říká programu SIPp, že zpráva bude obsahovat informace nutné pro následné autentizování, a pole

crlf="true" je pouze doplňujícím polem, které zaručí, že ve výpisu programu na obrazovce bude mezi touto a následující zprávou vynechán řádek, slouží tedy ke zpřehlednění.

Dále scénář obsahuje kromě odesílaných a přijímaných zprávy i část říkající, kdy a jak dlouho mají být přehrávána média. Toto obstarává následující blok kódu:

```
<nop>
<action>
<exec play_pcap_audio="/home/loki/sipp/g711u.pcap"/>
</action>
</nop>

<pause milliseconds="60000" crlf="true"/>
```

Vyhledávání informací uvnitř hlaviček, které je nezbytné při testování SIP Proxy se pak provede následujícím blokem kódu:

```
<action>
<ereg regexp="[1-9].*[^>]" search_in="hdr"
header="Contact:" check_it="true" assign_to="1" />
</action>
```

Tento kód porovnává pole *Contact:* příchozí SIP zprávy s regulárním výrazem *"[1-9].*[^>]"*, tedy s číslicí 1-9 následovanou libovolným počtem libovolných znaků, přičemž ukončujícím znakem je >, který ale nebude do výběru zahrnut. Výsledkem je vyextrahování IP adresy, portu a transportního protokolu z pole *Contact:* a jeho přiřazení k proměnné označené jako „1".

Poslední částí scénáře je definice časových intervalů. Tuto funkci plní dvě pole, která obsahují časy v milisekundách a vypadají následovně:

```
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100"/>

<CallLengthRepartition value="10, 100, 1000, 10000"/>
```

První definuje časové intervaly, kterým se budou porovnávat časovače definované scénářem. Druhé je pak stejným způsobem aplikováno na hovory. Ve výsledných datech tedy nebude uveden přesný čas každého jednotlivého časovače/hovoru, ale počet měření časovače/hovorů, které se vejdou do intervalu definovaného sousedními časy. Výsledky tedy budou například obsahovat údaj, že 50 hovorů trvalo od 0 do 10 ms, 75 hovorů od 10 do 100 ms, atd. Pro zvýšení přesnosti je nutné dané časy definovat precizněji.

Scénář pro UAC je uložen ve složce */home/loki/sipp/* pod jmény, která odpovídají danému scénáři:

- *uac_proxy.xml*,
- *uac_transcoding_g711u_g711u.xml,*
- *uac_transcoding_g711u_g711A.xml,*
- *uac_transcoding_g711u_gsm.xml,*
- *uac_transcoding_g711u_g726.xml.*

Hodnoty a kompletní soubory čtenář nalezne na přiloženém DVD.

6.2.3 Konfigurace a příprava SIPp pro fungování jako UAS

SIPp v režimu UAS nemá pouze nižší nároky na hardwarový výkon počítače, je i jednodušší na konfiguraci. V tomto režimu SIPp nepotřebuje soubor s médii, jelikož je pouze vrací zpět, dále soubor *csv* je v podstatě totožný se souborem určeným pro UAC s tím rozdílem, že třetí sloupec obsahující číslo volaného je zbytečný, a proto je ponechán prázdný.

I scénář má podobnou strukturu jako scénář určený pro UAC, avšak je třeba zde mít scénáře dva. První, který zaregistruje všechny dostupné UAS na serveru a druhý, který bude poslouchat na stanoveném portu příchozí SIP žádosti. Registrace je totožná s UAC, druhý scénář je pak podobný, avšak objevují se zde další pole interně doplňovaná programem SIPp. Jedná se o pole *last_via*, *last_from*, apod. Jejich význam je ale zřejmý z jejich názvu, neboť pouze extrahují to které pole z hlavičky příchozí SIP zprávy, a proto je zde dále nebudu rozebírat.

Všechny soubory jsou opět uloženy ve složce */home/loki/sipp/* a pod jmény *csv2.csv, srs.xml, uas_proxy.xml* a *uas_transcoding_g711u_gsm.xml*.

6.3 SAR

Jelikož nás při testech nejvíce zajímá, jaké je vytížení hardwaru ústředny, je potřeba mít k dispozici program, který dokáže toto vytížení měřit. Takovým programem je *SAR*, neboli System Activity Reporter. Jedná se o součást balíku systémových nástrojů, který je v repozitářích Debianu uložen pod souhrnným názvem *SYSSTAT*.

Instalace tohoto programu je triviální, jen je třeba dbát na to, aby SIP server, na kterém budeme pomocí SARu měřit vytížení, měl stejnou verzi balíku SYSSTAT jako počítač, na kterém budeme data vyhodnocovat. Instalaci provedeme prostým:

```
# aptitude install sysstat
```

Program SAR se spouští z příkazové řádky, přičemž jednotlivé možnosti a přepínače tohoto programu jsou uvedeny v manuálu[12] a pro jejich trivialitu se jimi dále nebudeme zabývat. Skriptový příkaz, kterým SAR spouštíme, bude uveden dále v podkapitole testování.

6.4 Asterisk

Pobočková ústředna Asterisk je zahrnuta v repozitářích Debianu, takže její instalace je opět triviální, jak ukazuje následující příkaz. Avšak než takto nainstalujeme Asterisk, je třeba si ověřit, jaká verze je zrovna v repozitářích dostupná. V době vzniku této práce to byla verze 1.4, která je ovšem poměrně zastaralá, proto je třeba najít balíčky obsahující novější verzi 1.6. Tyto balíčky jsou dostupné pro „testing“ verzi Debianu (Squeeze), takže stačí pouze modifikovat soubor */etc/apt/sources.list* a následně můžeme nainstalovat Asterisk ve verzi 1.6. Do zmiňovaného souboru přidáme řádek:

```
deb http://ftp.cz.debian.org/debian squeeze main
```

Instalaci pak provedeme klasicky pomocí:

```
# aptitude install asterisk
```

Instalací získáme plně funkční SIP server, který je ale třeba nakonfigurovat. Vzhledem k proklamované snaze o co největší jednoduchost budou úpravy poměrně malé a soustředí se pouze na soubory */etc/asterisk/extensions.conf* a */etc/asterisk/sip.conf*.

6.4.1 Úprava souboru sip.conf

Tento soubor v Asterisku definuje parametry jednotlivých uživatelů a jejich účtů. Vzhledem k potřebám této práce bude potřeba, aby tento soubor obsahoval velké množství účastníků, na konce již existujícího souboru přidáme 8 000 bloků informací podobných tomuto:

```
[50000]
type=friend
context=test
host=dynamic
secret=uas
canreinvite=no
nat=no
disallow=all
allow=ulaw
```

Tímto blokem jsme přidali do databáze Asterisku uživatele se jménem „50000“ a heslem „uas“. Dále jsme zabránili tomu, aby se Asterisk pokoušel vytvořit přímé spojení mezi klienty (tzv. RTP bridge) a povolili jsme pro tohoto uživatele jediný kodek a to G.711μ. Změnou kodeku v posledním řádku přinutíme Asterisk, aby přistoupil k translaci kodeků, ovšem pouze za předpokladu, že na obou koncích spojení budou použity (nastaveny) různé kodeky. Podrobnější vysvětlení jednotlivých hodnot lze najít například na stránkách společnosti Telegro[13].

6.4.2 Úprava souboru extensions.conf

V tomto souboru najdeme pravidla, která se mají aplikovat v případě, že na Asterisk přijde žádost o vytočení některého čísla. Je mimo rámec této práce podrobně rozebrat možnosti a způsoby nastavení tohoto souboru. Pro zprovoznění Asterisku tak, aby spojoval uživatele, které simuluje SIPp, nám postačí následující řádky.

```
[test]
exten => _[1-9]XXXX,1,Dial(SIP/${EXTEN},25,r)
```

Tento dialplan nám říká, že pokud se někdo bude pokoušet spojit s pětimístným číslem nezačínajícím nulou, pak se toto číslo okamžitě vytočí. Tento dialplan je velice jednoduchý a to zejména kvůli základní filosofii našeho testování, kdy není možné otestovat veškeré možnosti a nastavení Asterisku, proto výsledky měření vzešlé z testů na tomto dialplanu lze brát za limitní hodnoty výkonu Asterisku, přičemž v reálném provozu bude jeho výkon vždy nižší.

6.5 Opensips

Instalace Opensips je obdobná instalaci Asterisku. Tentokrát ovšem nemůžeme sáhnout do repozitářů Debianu, ale musíme použít balíčky vytvořené přímo výrobcem programu Opensips. Do souboru */etc/apt/sources.list* dopíšeme tyto řádky:

```
deb http://debian.leurent.eu/debian stable main
deb-src http://debian.leurent.eu/debian/ stable main
```

Po následném provedení updatu databáze můžeme Opensips nainstalovat. K fungování Opensips budeme potřebovat některou z databází. V době psaní této práce byl Opensips plně funkční pouze s databází MySQL, proto byla použita tato databáze. Při instalaci budeme upozorněni na neexistenci klíče potvrzujícího identitu zdroje balíčku, této chybové informaci si ale nemusíme všímat, neboť nemá vliv na instalaci, kterou provedeme následujícími příkazy:

```
# aptitude install opensips
# aptitude install opensips-mysql-module
```

Vzhledem k závislosti programu Opensips na databázi MySQL je potřeba mít nainstalován taktéž MySQL server.

Konfiguraci programu řeší soubor */etc/opensips/opensips.cfg*, drobné změny je však třeba provést i v souborech */etc/default/opensips* a */etc/opensipctlrc*.

6.5.1 Konfigurace Opensips

Nejprve je třeba povolit spuštění programu, to se provede v souboru */etc/default/opensips*, kde je potřeba u možnosti RUN_OPENSIPS dát za rovná se „yes“. V tomto souboru je možné ovlivnit i množství sdílené paměti, pro testovanou hardwarovou konfiguraci bylo použito 1 024 MB.

V souboru */etc/opensipctlrc* je pak nutné odkomentovat následující pole:

- SIP_DOMAIN=10.0.0.140,
- DBENGINE=MYSQL,
- DBHOST=localhost.

Ostatní řádky je možné nechat beze změny. Adresa uvedená u domény je adresou, kterou ve všech našich testech bude mít síťová karta SIP serveru.

Hlavní konfigurace se provádí v souboru */etc/opensips/opensips.cfg*. Vyjdeme z defaultní varianty tohoto skriptu a provedeme jen několik dílčích úprav.

Na začátku snížíme množství informativních zpráv, které nám Opensips zobrazuje. Napíšeme tedy „debug=1“, což omezí výpisy pouze na chybová hlášení. Hodnota uvedená u pole „children“ je určující pro množství podprocesů, na které se může Opensips rozdělit. Pro měření budeme používat dvě možnosti – 4 a 16. Následně odkomentujeme tyto řádky, čímž zabezpečíme to, že se Opensips nebude snažit překládat IP adresy na doménová jména a naopak, a také umožníme použití databáze, vůči které budeme provádět autentizaci.

```
auto_aliases=no
listen=udp:10.0.0.140:5060
loadmodule „db_mysql.so“
loadmodule „auth.so“
loadmodule „auth_db.so“
```

V další části konfiguračního skriptu je třeba upravit bloky zabývající se prací s databází, takže ve skriptu budou tyto bloky textu:

```
# ----- usrloc params -----

#modparam("usrloc", "db_mode",   0)
/* uncomment the following lines if you want to enable DB
persistency for location entries */
modparam("usrloc", "db_mode",   0)
modparam("usrloc", "db_url",
"mysql://root:root@localhost/opensips")


# ----- auth_db params -----

/* uncomment the following lines if you want to enable the
DB based authentication */
modparam("auth_db", "calculate_ha1", 1)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "db_url",
"mysql://root:root@localhost/opensips"
modparam("auth_db", "load_credentials", "")
```

Poslední úprava má za cíl umožnit autentizaci účastníků vůči databázi a údajům v ní. Abychom toto umožnili, je třeba okomentovat dva bloky textu, přičemž první se vztahuje k žádostem jiným než je zpráva REGISTER, zatímco druhá řeší právě tyto zprávy.

```
if (!(method=="REGISTER") && from_uri==myself)
     {
          if (!proxy_authorize("", "subscriber")) {
               proxy_challenge("", "0");
               exit;
          }

          if (!db_check_from()) {
               sl_send_reply("403","Forbidden auth ID");
               exit;
          }

          consume_credentials();
     }


if (is_method("REGISTER"))

     {
          if (!www_authorize("", "subscriber"))
          {
               www_challenge("", "0");
```

```
            exit;
        }

        if (!db_check_to())
        {
            sl_send_reply("403","Forbidden auth ID");
        exit;
        }

        if (!save("location"))
        sl_reply_error();

exit;
}
```

Těmito úpravami jsme nastavili Opensips tak, aby fungoval jako jednoduchá SIP Proxy, která bude kontrolovat přihlašovací údaje a spojovat hovory v rámci jedné domény. Abychom však mohli začít testovat, musíme vytvořit databázi a vložit do ní data, která obsahují údaje o uživatelích, to se provede těmito příkazy:

```
# opensipsdbctl create
# opensipsctl add uživatel heslo
```

První příkaz vytvoří databázi MySQL s názvem „opensips“, druhý do ní vloží uživatele se jménem „uživatel“ a heslem „heslo“. Vzhledem k výkonu SIP Proxy je takovýchto uživatelů potřeba několik desítek tisíc (v našem případě 72 000), proto je vhodné vkládání uživatelů ošetřit skriptem a neprovádět jej manuálně.

7 REALIZACE TESTŮ

Nyní, když víme, kterak nakonfigurovat jednotlivé počítače tak, aby v síti fungovaly jako UAC i UAS a aby na SIP serveru byla monitorována hardwarová zátěž, můžeme přistoupit k samotným testům. Veškeré testování je automatizováno a je prováděno prostřednictvím bashového skriptu, který není nikterak složitý, proto si zde rozebereme jen jeho nejdůležitější prvky. Kompletní verzi skriptu pro testování jak B2BUA, tak SIP Proxy čtenář najde na přiloženém DVD.

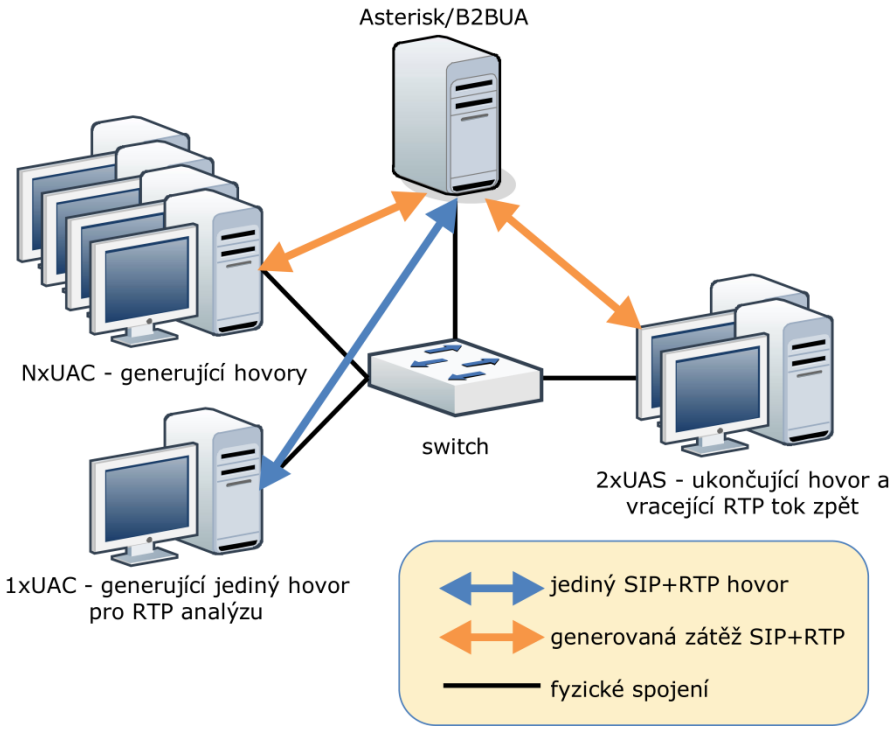
Další problematikou, kterou zde budu rozebírat jen okrajově, je adresace sítě, neboť ta se může lišit případ od případu, aniž by došlo k ovlivnění výsledků, nebo testovací metodiky. Adresy budou pro oba případy uvedeny jen informativně a pouze pro orientaci čtenáře v daném textu a příkazech.

7.1 Testy B2BUA

B2BUA má oproti SIP proxy v **základní konfiguraci** význačnou vlastnost – prochází přes něj média. Již víme, že SIP server, který bude vystupovat jako B2BUA bude Asterisk, který bude vystupovat jako prostředník mezi komunikujícími SIPp. Zároveň na něm bude docházet k translaci kodeků v případě, že každý z konců komunikace bude používat jiný kodek. Této vlastnosti se dá využít pro vytvoření univerzální testovací metodiky, která umožní porovnávat B2BUA nezávisle na zvolené platformě – hardwarové i softwarové (viz kap. 9).

Pomocí SIPp tedy budeme generovat hovory, které budou obsahovat 60 sekund hlasových dat. Toto bude obstarávat SIPp v roli UAC. Na druhé straně bude SIPp v roli UAS přijímat odeslané zprávy i RTP pakety nesoucí hovorová data, přičemž tyto RTP pakety bude vracet nezměněné zpět. Tímto způsobem vygenerujeme zátěž, kterou budeme postupně zvyšovat. Dále pak budeme sledovat jak hardwarové vytížení ústředny, tak i zmiňované RRD a SRD a dále pak i kvalitu hovoru kvantifikovanou hodnotami Jitteru a zpoždění mezi pakety. Úspěšnost měření pak budeme posuzovat i z hlediska procenta úspěšných registrací/hovorů. Jelikož je B2BUA zatěžován v průběhu celého hovoru, je logické vzít jako metriku množství současných hovorů. Vůči této metrice pak budeme vztahovat veškeré naměřené parametry.

Topologie pro měření B2BUA je zobrazena na obr. 7.1, který se od dříve uvedené topologie liší pouze uvedením IP adres a naznačením toku SIP signalizace a médií. Je zde taky uveden konkrétní počet UAC, neboť pro otestování ústředny s dříve uvedenými parametry stačily 4 počítače simulující klientský konec komunikace. Pro měření kvality hovoru je zde využit samostatný počítač, který do komunikace vstupuje uprostřed testu, kdy vygeneruje pouze jediný hovor, který je pak prostřednictvím Wiresharku zachytáván a následně i analyzován.



Obr. 7.1: Topologie pro měření B2BUA s vyznačenými toky zpráv a médií

Z pohledu testování je nejprve nutné zaregistrovat SIPp, která vystupují jako UAS, a to z toho důvodu, aby B2BUA věděl, kam má požadavky klientů směrovat. Registraci UAS provedeme spuštěním následujícího příkazu:

```
ssh root@10.0.0.$a sipp -sf /home/loki/sipp/srs.xml -inf
/home/loki/sipp/csv2.csv 10.0.0.140 -i 10.0.0.$a -m 2001 -r 100
-bg &
```

Tento příkaz se přihlásí prostřednictvím SSH na počítač, který má simulovat UAS, přičemž znak „$a“ zastupuje poslední číslo IP adresy, které je doplněno až prostřednictvím skriptu. V našem případě zde budou použity 2 hodnoty a to 141 a 142. IP adresa, která je uvedena bez přepínače (tedy 10.0.0.140) je adresa serveru, vůči kterému se UAS registruje. Další přepínače si rozebereme v odrážkách:

- sf – odkaz na soubor se scénářem,
- inf – odkaz na soubor s informacemi o hovoru,
- m – maximum hovorů, kterého chceme dosáhnout,
- r – množství hovorů generovaných za jednu sekundu (nebo dobu uvedenou a přepínačem -rp),
- bg – SIPp se spustí na pozadí.

Ampersand na konci příkazu je nutnost z hlediska skriptu, ze kterého tento příkaz spouštíme, jelikož počítači říká, aby s vykonáváním dalších příkazů nečekal na ukončení spuštěného SIPp.

Jakmile je proces registrace ukončen, je potřeba spustit SIPp v režimu UAS.

```
ssh root@10.0.0.$a sipp -sf /home/loki/sipp/uas_transcoding
_g711u_gsm.xml -i 10.0.0.$a -rtp_echo -nd -bg &
```

Logika tohoto příkazu je analogická té předchozí. Přibyl ale přepínač „rtp_echo“, který zajistí vracení RTP paketů zpět ke zdroji, tedy UAC, a přepínač „nd“, který zakáže některé vlastnosti SIPp, jako například ukončování hovoru po přijetí neočekávané zprávy.

Tímto jsme spustili SIPp v režimu UAS a můžeme přejít ke spuštění UAC. Zde je registrace součástí každého hovoru, proto ji není třeba řešit samostatným příkazem. Co je však důležité je určení doby, po kterou bude UAC generovat hovory. Jako optimální kompromis mezi časovou náročností a nutností generovat co nejdéle, aby se mohly objevit i hůře detekovatelné chyby, jsem zvolil 15 minut. Po tuto dobu ale nebude množství hovorů konstantní, konstantní bude až od druhé do patnácté minuty.

SIPp jako UAC spustíme příkazem:

```
ssh root@10.0.0.${UAC[$b]} sipp -sf /home/loki/sipp/$jmeno.
xml -inf /home/loki/sipp/csv.csv 10.0.0.140 -i 10.0.0.${UAC[$b]}
-r $n -rp 4s -m ${max2} –timer _resol 50 -t un -trace_screen -
trace_stat -trace_rtt -nd -l 5000 -timeout 1000s -bg &
```

Opět je logika totožná s tou uvedenou u registrace UAS, rozdíl je v tom, že skript mění jak množství generovaných hovorů za 4 sekundy (přepínač „r“ a „rp“) tak i celkové množství hovorů, které budou za dobu testu vygenerovány (přepínač „m“), proto u nich není uvedena konkrétní hodnota, ale zástupné proměnné, které si skript sám vypočítává. Hodnota přepínače „r“ je volena v závislosti na množství současných hovorů, kterého chceme dosáhnout. Lze jej určit z rovnice:

(4) $$r = k \cdot p$$ [hov.s^{-1}; hov.s^{-1},-]

kde $$k = N/t$$ [hov.s^{-1}; hov.,s]

Význam jednotlivých veličin je tento:

- N je počet současných hovorů, které chceme generovat
- p – je perioda (v našem případě 4 s)
- t – je délka jednoho hovoru

Například pokud chceme generovat 120 současných hovorů, pak do pole přepínače „r“ musíme vložit číslo 8, neboť vzhledem k délce hovoru rovné 60 sekundám potřebujeme vygenerovat 2 hovory za sekundu tak, aby po 60 sekundách testu bylo vytvořeno celkem 120 hovorů (60*2=120). Avšak nemáme periodu rovnu 1 sekundě, ale 4 sekundám, proto je nutné číslo 2 ještě vynásobit 4, čímž získáme číslo 8. Čtyřsekundová perioda je zvolena z důvodu umožnění menšího kroku, než je 60 hovorů. Dále je nutné brát ohled ještě na množství zapojených počítačů, ale to už pro jednoduchost nebudeme rozebírat. V případě zájmu čtenáře o to, jakým způsobem jsou vypočítávány jednotlivé proměnné, nechť se tento podívá na skript na přiloženém DVD.

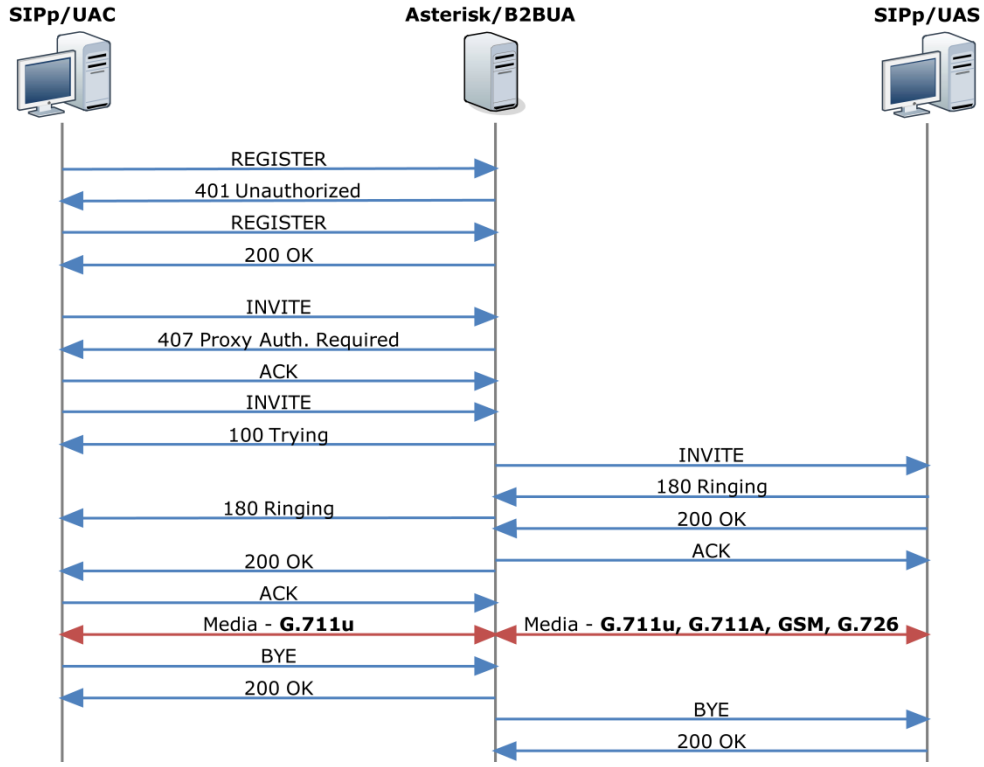
Přepínače „trace…“ slouží ke spuštění generování souborů obsahujících data z měření, pro vyhodnocení měření ve stylu této práce je možné přepínač „trace_screen“ vypustit, neboť jím generovaná data jsou obsažena i v souborech vygenerovaných pomocí zbylých dvou přepínačů.

V použité topologii byly jako pro UAC alokovány IP adresy 10.0.0.144 – 148, neboť pro vygenerování dostatečného zatížení postačovaly 4 PC.

Ostatní přepínače pak mají tyto funkce:

- timer_resol – nastavuje preciznost vnitřního časovače SIPp (má velký vliv na výkonnost SIPp),
- t – přepínač definující transport; možnost „un“ říká SIPp, aby generovalo každý nový hovor z jiného portu UDP, čímž simuluje více jednotlivých volajících a více tak odráží reálné prostředí,
- l – definuje maximum současných hovorů (není vhodné jej použít pro definici, neboť ovlivňuje množství generovaných hovorů za sekundu),
- timeout – ukončí SIPp po specifikovaném čase i v případě nedokončených hovorů.

Grafickou podobu scénářů pro UAC i UAS v případě měření na B2BUA zobrazuje obr. 7.2.



Obr. 7.2: Tok SIP zpráv při měření na B2BUA

Posledním příkazem, který je třeba si rozebrat je spuštění SARu, to se provede následovně:

```
ssh root@10.0.0.$PBX sar -u -r -P ALL -n DEV -o
/sar/vysledky_$jmeno/data_$pocet.sar 10 60 >/dev/null 2>&1
```

Tímto způsobem spustíme SAR na pozadí tak, aby co 10 sekund nasbíral data o paměti, CPU i síťových rozhraních a tato data uložil do binárního souboru, jehož název obsahuje počet hovorů generovaných za sekundu. Tento soubor je pak nutné převést do čitelné podoby, což provede příkaz:

```
sar -u -r -P ALL -n DEV –f /home/loki/sipp/${jmeno}/sar/
data_${pocet}.sar > /home/loki/sipp/${jmeno}/sar/data_${pocet}.
txt
```

Tento příkaz je však již spouštěn až na hlavním UAC. Jelikož jsou data sbírána periodicky (perioda=p_{SAR}) a celkově je sbíráno N_{SAR}=60 vzorků, pak celková doba měření (T_{SAR}) vyplývá ze vztahu:

$$T_{SAR} = N_{SAR} \cdot p_{SAR} \quad [s;-,s] \tag{5}$$

Jednoduchým dosazením zjistíme, že data jsou SARem sbírána 10 minut a to z toho důvodu, aby zaznamenané výsledky byly nasbírány v době, kdy je zatížení generované SIPp konstantní. Stejným způsobem je třeba ošetřit data nasbíraná prostřednictvím SIPp a to zejména data časovačů, toto je ale možné provést až při vyhodnocování.

7.2 Testy SIP Proxy

Na rozdíl od B2BUA není u SIP proxy možné využít procházejících médií, proto se v tomto případě omezíme na měření draftových RRD a SRD, hardwarového vytížení a úspěšnosti registrací a hovorů.

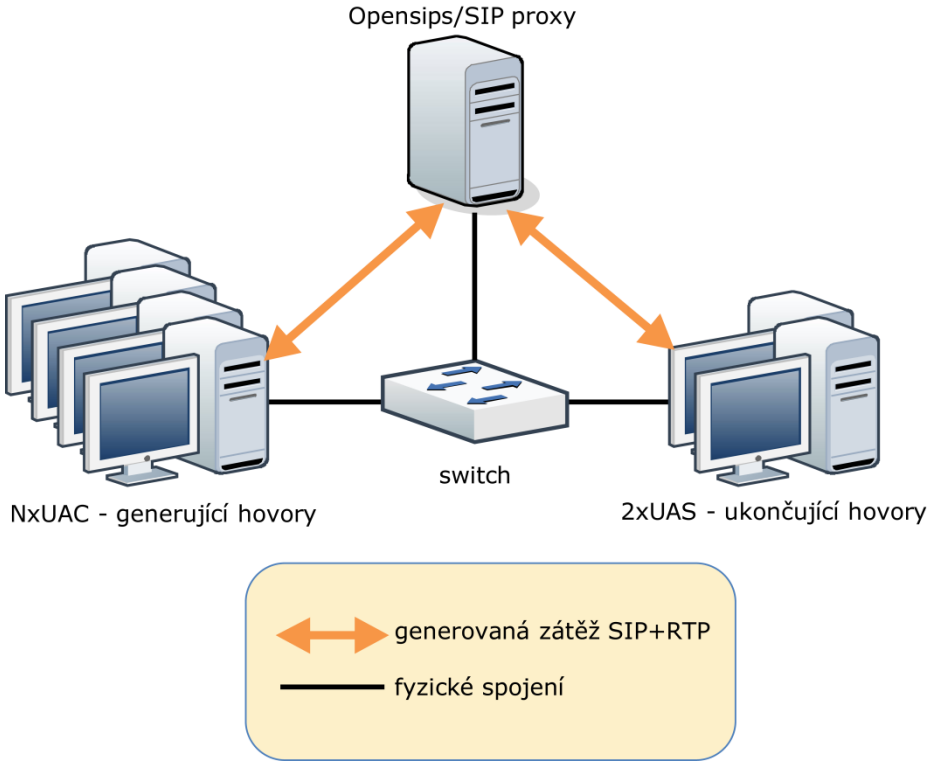
Vzhledem k obsáhlosti podkapitoly věnované B2BUA si zde projdeme jen rozdíly oproti B2BUA, kterých ovšem není mnoho.

Opět budeme 15 minut generovat hovory pomocí SIPp v režimu UAC a tyto hovory budeme přes SIP proxy směrovat na SIPp v režimu UAS. Spuštění všech prvků je totožné jako v případě testování B2BUA, rozdíl najdeme pouze ve scénáři pro UAC. Jednak zde nejsou přítomna média, neboť média neprocházejí ústřednou, a proto je zbytečné je generovat, a jednak dochází k vyhodnocení příchozích SIP hlaviček na straně UAC, které si bere informace pro přímé směrování zpráv na UAS. SIP proxy je v cestě udržena prostřednictvím Record-route.

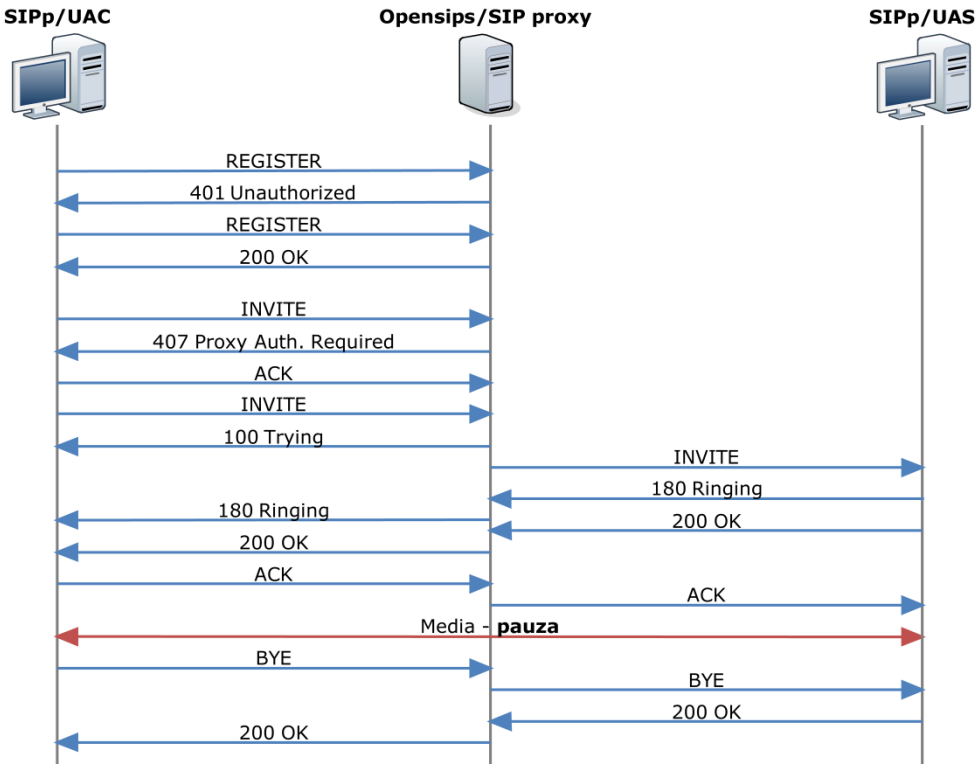
Jelikož je SIP proxy vytěžována pouze signalizací, nemá smysl brát jako metriku množství současných hovorů, daleko smysluplnější je v tomto případě hodnota množství hovorů generovaných za jednu sekundu, což bude také největší rozdíl mezi grafy pro B2BUA a SIP proxy.

Adresace byla obdobná té použité pro B2BUA s tím rozdílem, že UAC bylo použito 12 s IP adresami 10.0.0.144-155.

Testovací topologii pro měření na SIP proxy ukazuje obr. 7.3 a grafická podoba scénářů pro UAC i UAS v případě měření na SIP proxy je pak zobrazena na obr. 7.4.



Obr. 7.3: Topologie pro měření SIP proxy s vyznačenými toky zpráv



Obr. 7.4: Tok SIP zpráv při měření na SIP proxy

8 VYHODNOCENÍ VÝSLEDKŮ TESTŮ

Výsledkem měření, ke kterému celá tato práce směřovala, je velké množství dat uložených v mnoha souborech. Pro jejich zpracování je proto výhodné napsat si skript. Tento skript byl vytvořen i pro tuto práci a čtenář jej najde na přiloženém DVD. Jazyk, ve kterém byl tento skript napsán, je BASH, tedy varianta shellu používaná v Debianu a Ubuntu. Ačkoliv základem je tedy shell, hlavní úlohy zpracovává jazyk AWK[14], který je určen pro správu textových souborů. Výsledkem zpracování výsledků zmiňovaným skriptem je textový soubor obsahující přehledné tabulky pro jednotlivé kroky v měření. Tento textový soubor je ve formátu čitelném programem MS Excel, odkud je možné exportovat data do jiných programů.

Množství celkových výsledků je také důvodem, proč zde, v hlavním textu této práce, bude uveden postup vyhodnocení výsledků pouze pro případ měření translace kodeků na SIP serveru nakonfigurovaném do režimu B2BUA. A ani tyto výsledky nebudou kompletní, zobrazeny budou pouze grafické výsledky měření efektivity translace z G.711u do GSM. Kompletní výsledky budou v grafické podobě prezentovány v tištěných přílohách, odkud si čtenář může odvodit závěry uvedené v poslední kapitole. Má-li čtenář zájem o prostudování jednotlivých číselných dat, nebo hlasových souborů, může si tyto najít na přiloženém DVD.

Tato kapitola tedy bude obsahovat praktickou aplikaci navržené metodiky, příklad, jak postupovat při vyhodnocování výsledků získaných při měření postupem uvedeným v dřívějších kapitolách této práce.

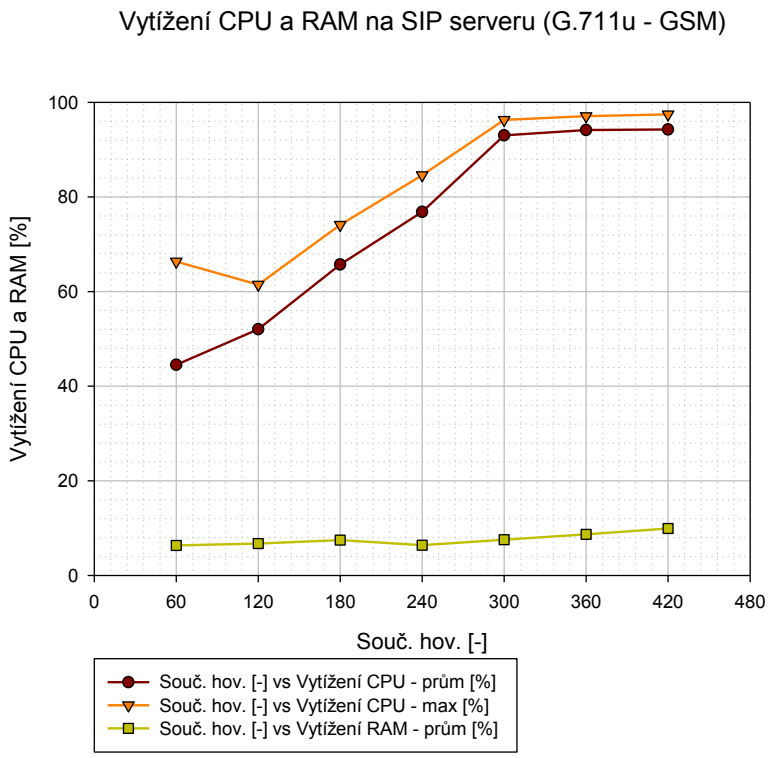
8.1 Analýza výsledků zatížení SIP serveru a kvality hovoru

Jako první si rozebereme výsledky naměřené programem SAR, to proto, že tyto výsledky jsou vždy nejsnáze dostupné (třeba výpisem hardwarové utilizace programem TOP) a hlavně nejvíce ovlivňují naše rozhodování o tom, pro jaké zátěže je SIP server ještě svým výkonem postačující. Uvedené výsledky ukazují, jak již bylo řečeno, měření translace z G.711u do GSM. Obr. 8.1 ukazuje rostoucí zatížení procesoru a paměti v závislosti na množství současných hovorů.

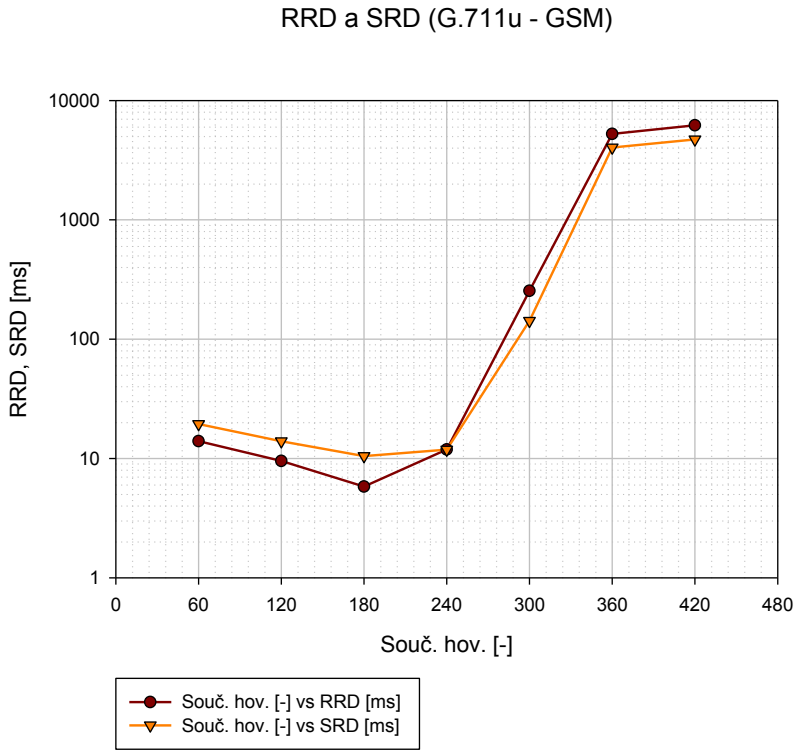
Z tohoto grafu je patrné, že zatímco paměť je využita relativně málo a její utilizace s množstvím současných hovorů téměř neroste, naproti tomu však využití CPU roste naprosto zásadně. Odsud lze odvodit, že v případě B2BUA nastaveného na translaci kodeků je klíčový výkon procesoru. Pro náš případ je tedy SIP server schopen **efektivně zvládnout zhruba 300 hovorů**.

Nyní se podíváme na výkon B2BUA z pohledu SIPu. Jak bylo popsáno v dřívějších kapitolách, využijeme k tomu charakteristiky RRD a SRD. Obr. 8.2 pak ukazuje graf zjištěných hodnot opět v závislosti na množství současných hovorů. Vzhledem k velkým rozdílům mezi hodnotami bylo využito logaritmické měřítko. Při získávání dat pro tento graf je třeba mít na paměti, že data získaná pomocí SIPp v sobě nesou chybu způsobenou tím, že byla sbírána i v době, kdy vytížení nedosahovalo uvedené úrovně. Z tohoto důvodu je vhodné vybírat data až po uplynutí 2,5 minut po startu testu a ukončit jejich sběr po 10-ti minutách. SIPp tuto volbu neumožňuje, proto jsem tento výběr ošetřil generujícím skriptem prostřednictvím jazyka AWK (obsažen na DVD). Jelikož z pohledu SIP transakcí je zpoždění problematické, dosáhne-li řádu

stovek ms, můžeme usoudit, že z tohoto pohledu je maximální snesitelnou zátěží opět **300 hovorů**. Pokles v oblasti okolo 180-ti současných hovorů je způsoben proměnlivostí měřených hodnot, přičemž zátěž nedosahuje úrovně, při které by měřitelně zvýšila dané parametry.

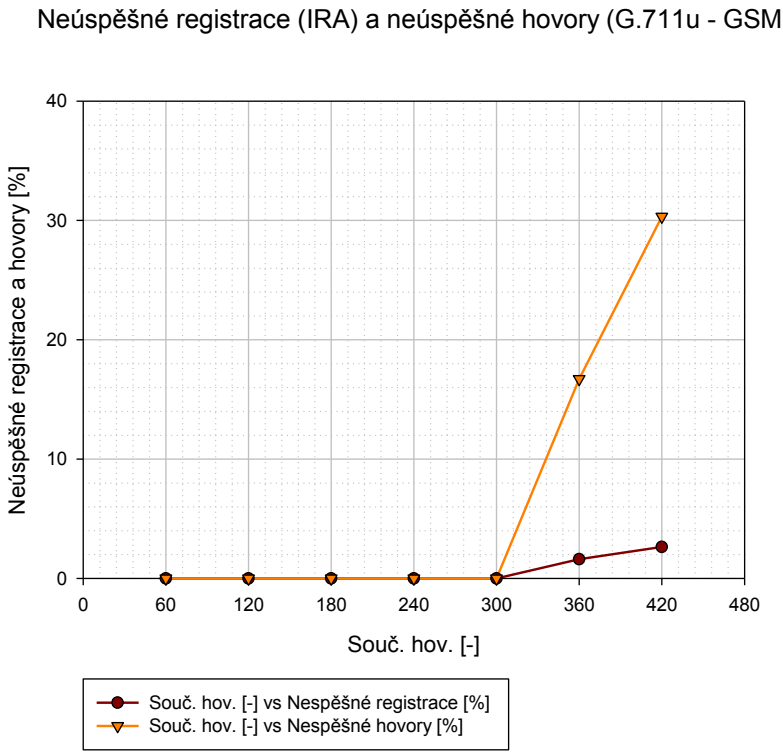


Obr. 8.1: Zatížení CPU a paměti v závislosti na množství současných hovorů (G.711u – GSM)



Obr. 8.2: RRD a SRD (G.711u – GSM)

Dalším hlediskem, ze kterého budeme posuzovat schopnosti SIP serveru, je množství neúspěšně odbavených registrací (IRA) a hovorů, jejichž graf zobrazuje obr. 8.3. Z tohoto grafu je opět patrné, že z pohledu registrací i hovorů je možné danou ústřednu používat **do zatížení 300 současnými hovory**. Limity pro toto kritérium byly stanoveny dle obecně platných podmínek užívaných v telekomunikační praxi, kdy je maximální snesitelná neúspěšnost hovorů stanovena na 1%. Vzhledem k následnosti registrace a hovoru v použitém scénáři je nutné množství neúspěšných hovorů vztahovat ne k celkovému počtu vygenerovaných hovorů (registrací a hovorů), ale pouze k množství úspěšných registrací, aby se množství neúspěšných registrací neobjevilo ve statistikách dvakrát.

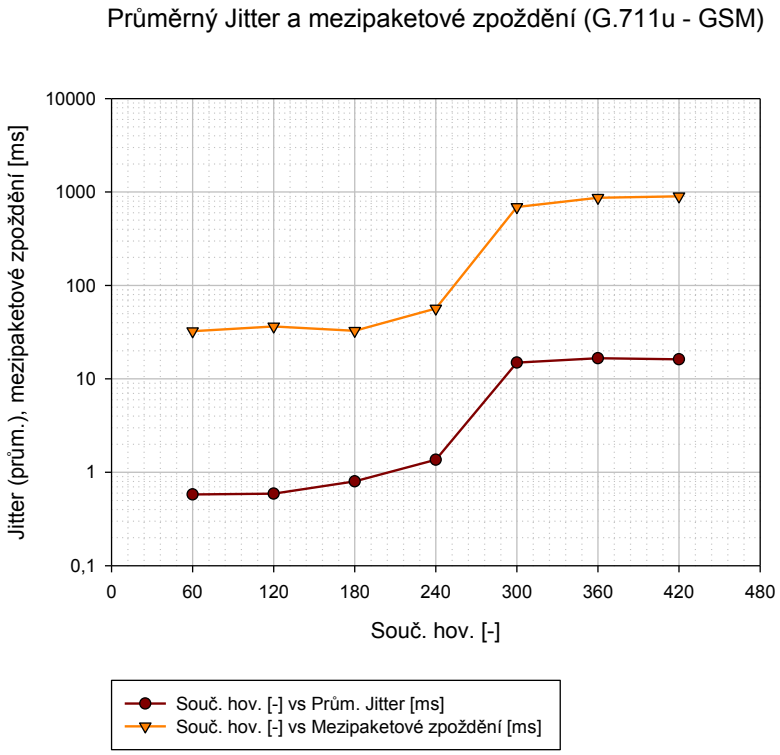


Obr. 8.3: Neúspěšné registrace a hovory (G.711u – GSM)

Poslední samostatnou statistikou bude vyhodnocení dynamických parametrů SIP serveru z hlediska RTP. Dynamickými parametry, které budeme sledovat, jsou Jitter, tedy variabilita zpoždění mezi sousedními pakety, a právě toto zpoždění. Na obr. 8.4 jsou shrnuty a graficky zpracovány výsledky zachycené programem Wireshark, tyto výsledky si lze ověřit, neboť zdrojové soubory jsou umístěny na přiloženém DVD. Z hlediska vyhodnocení těchto parametrů je třeba brát v úvahu, že ve Wiresharku jsou uvedeny dva datové toky, první od UAC k UAS (resp. B2BUA) a druhý ve zpětném směru. Jelikož SIPp v režimu UAC odesílá původní neovlivněný datový tok, musíme pro zjištění parametrů systému vyhodnocovat právě jenom zpětný směr, tedy od B2BUA k UAC, jehož data prošla celou soustavou a mohla tedy být ovlivněna parametry SIP serveru.

Pro RTP problematické zpoždění (od odeslání na jedné straně po přijetí na straně druhé) v řádech desítek ms, avšak námi uvedené charakteristiky definují zpoždění jiné, a sice zpoždění mezi jednotlivými pakety na přijímací straně. Ačkoliv se tedy jedná o dvě různá zpoždění, z hlediska významu pro hovor mají podobné důsledky, proto lze základní úvahu o zpoždění

hovoru aplikovat i na mezipaketové zpoždění, odkud plyne, že testovaný systém je možné z hlediska RTP používat do **zatížení** o velikosti **240 současných hovorů**.



Obr. 8.4: Průměrný Jitter a mezipaketové zpoždění (G.711u – GSM)

Ze všech dříve uvedených dynamických parametrů lze pak usuzovat na parametry statické, z nichž nejdůležitější je jednoznačně maximum současných hovorů, které lze SIP serverem odbavit. Jelikož v průběhu hovoru se projeví veškeré uvedené parametry, je nutné pro určení konečného verdiktu vzít ten s nejhorším výsledkem. V tomto případě je úzkým hrdlem zpracování RTP paketů v době jejich průchodu B2BUA, z hlediska tohoto kritéria, je systém použitelný pouze do přibližně **240 současných hovorů**.

Stejným způsobem lze posléze vyhodnotit výsledky měření i pro SIP proxy s tím rozdílem, že již nebude využito RTP. Vzhledem k této podobnosti zde již nebudu rozvádět vyhodnocení parametrů pro další z provedených měření, naproti tomu budou v závěru shrnuty výsledné hodnoty, které si pak čtenář může ověřit grafů v přílohách a z dat uložených na přiloženém DVD.

V případě B2BUA, jak již bylo nastíněno dříve, lze ale využít jeho základní vlastnosti, tj. průchodu médií SIP serverem, k definování metodiky pro obecné srovnání B2BUA napříč hardwarovými i softwarovými platformami. Tento postup si rozebereme nyní.

8.2 Efektivita translace kodeků jako obecný srovnávací parametr

Uvedené výsledky platí pouze pro daný SIP server v dané konfiguraci a není možné je nikterak porovnávat s jinými SIP servery a to z toho důvodu, že se může lišit architektura procesoru, pamětí, ale i samotného programu, čímž je měření logicky ovlivněno. Nelze tedy

stanovit obecnou rovnici, která by popisovala vztah výkonu SIP serveru například s frekvencí CPU nebo pamětí.

Je ovšem možné vzít si některý z naměřených výsledků jako standard a k němu vztáhnout výsledky všech ostatních měření, čímž se výsledné hodnoty očistí od závislosti na použité platformě a architektuře a bude možné porovnat například výkon Asterisku na CPU Intel a AMD při stejné frekvenci, nebo výkon Asterisku a Freeswitch na PC dané konfigurace. Uvedenou úvahu si rozebereme na již prezentovaných výsledcích pro translaci kodeků z G.711u do GSM. Jako základní hodnoty si zvolíme měření scénáře bez translace, tedy měření, kdy oba konce komunikace využívají stejný kodek G.711u.

Veškeré normalizované hodnoty, které zde budou uvedeny, byly získány měřením a zpracovány podle následující rovnice:

$$P_{RF} = \frac{P_{CT}}{P} \cdot 100 \qquad [\%; \%, \text{ms}] \tag{6}$$

P_{RF} je faktor efektivity translace, který je definován jako procentuální vyjádření podílu výkonu naměřeného v případě s translací (P_{CT}) a výkonu bez translace (P). Výkonem na pravé straně rovnice je některá z měřených veličin, tedy například hardwarová utilizace, nebo zpoždění, atd. Uvedenou rovnici lze tedy aplikovat na veškeré výsledky, proto jsou jednotky uvedeny nejednoznačně v procentech i ms, neboť je možné tuto rovnici vztáhnout například k procentuálnímu výkonu CPU, nebo ke zpožděním jako RRD, nebo SRD.

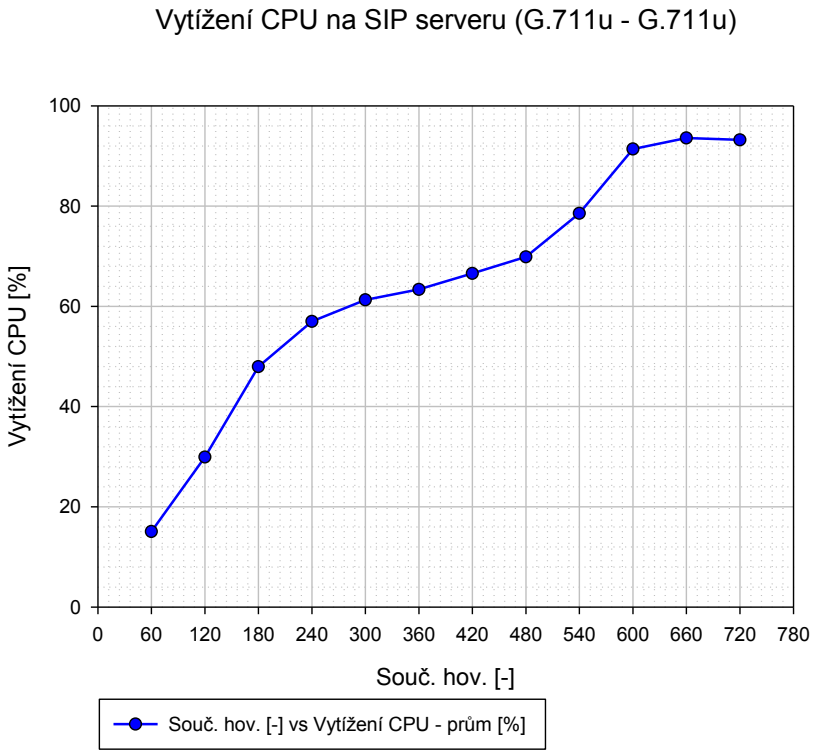
Jelikož je nutné vyjádřit výsledky vzhledem k základu naměřenému bez translace kodeků, budou následující grafy uvedeny vždy ve dvojicích – první graf (modrý) bude popisovat absolutní výkon naměřený bez translace kodeků, a druhý pak relativní hodnotu naměřenou při translaci vztaženou k případu bez translace. Na grafech je uveden pouze cílový kodek, není-li uveden, pak se jedná o výsledky měření bez translace kodeků. Prvním příkladem jsou obr. 8.5 a 8.6, které ukazují vytížení CPU.

Z druhého grafu je patrné, že pro nízkou zátěž 60-ti hovorů je translace kodeků z G.711u na GSM třikrát náročnější na výkon CPU, než prostý průchod hovorových dat bez translace. Následně pak normalizovaná zátěž klesá ke 140% vytížení bez translace, aby posléze opět rostlo. Pokles u konce druhého grafu je způsoben tím, že zatímco měření případu s translací již dosáhlo limitů systému, měření bez translace neustále roste. Uvedený graf je tedy hodnotný pouze do 300 současných hovorů.

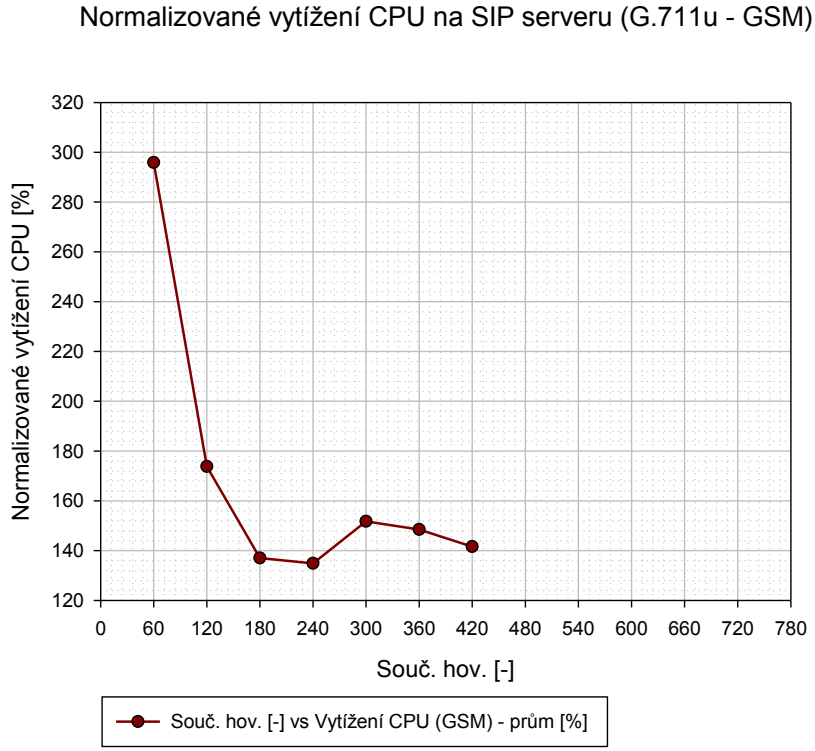
Následují grafy zaměřené na RRD a SRD (obr. 8.7 a obr. 8.8). Tyto charakteristiky jsou obzvláště zajímavé zejména z toho důvodu, že pro nízké zátěže jsou tato zpoždění dokonce menší než v měření bez translace. V některých případech jsou dokonce pouze 50%. Opět se jedná o typický problém způsobený značnou variabilitou těchto parametrů v případě, že zátěž ještě nedosahuje kritické úrovně. Následně pak ale obě charakteristiky strmě stoupají a při zátěži 300 současných hovorů se blíží 800násobku hodnot měření bez translace.

Poslední sada grafů (obr. 8.9 a 8.10) pak obsahuje hodnoty průměrného jitteru a mezipaketového zpoždění a jejich normalizované obdoby získané přepočtem naměřených hodnot podle rovnice (6). Obě normalizované hodnoty se pohybují okolo 100%, z čehož vyplývá, že hodnoty příslušných zpoždění jsou prakticky totožné jako hodnoty naměřené

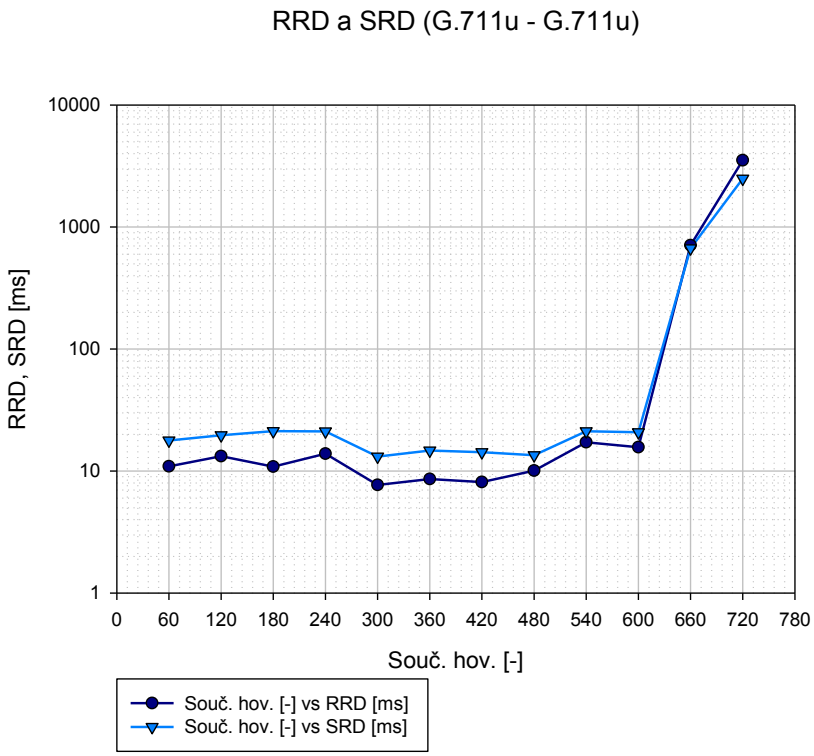
v případě bez translace. Mezi 240 a 300 hovory pak dochází ke strmému růstu obou normalizovaných hodnot na 11-ti násobek zpoždění bez translace kodeků.



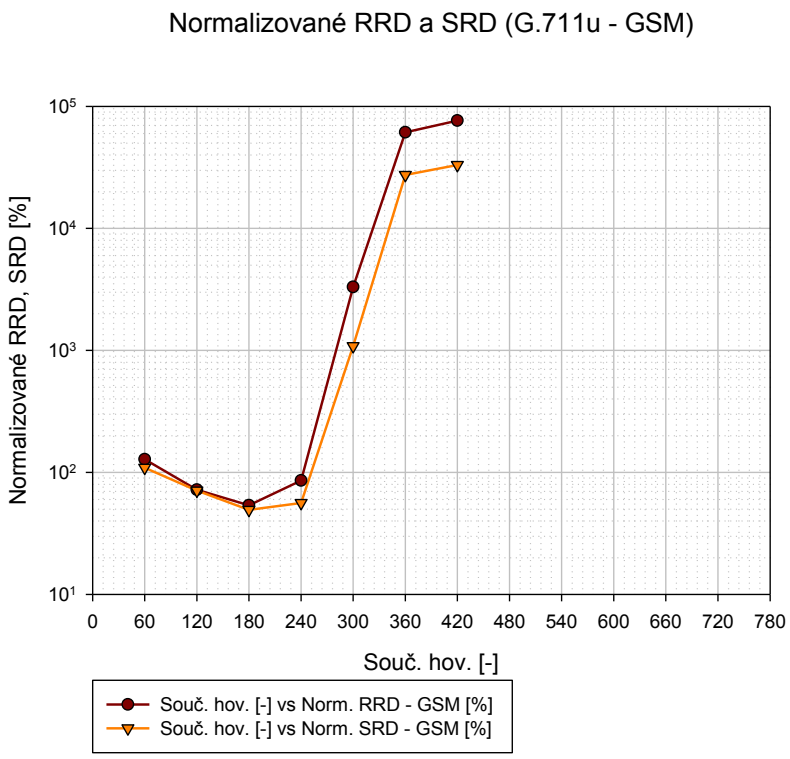
Obr. 8.5: Vytížení CPU na SIP serveru (G.711u – G.711u)



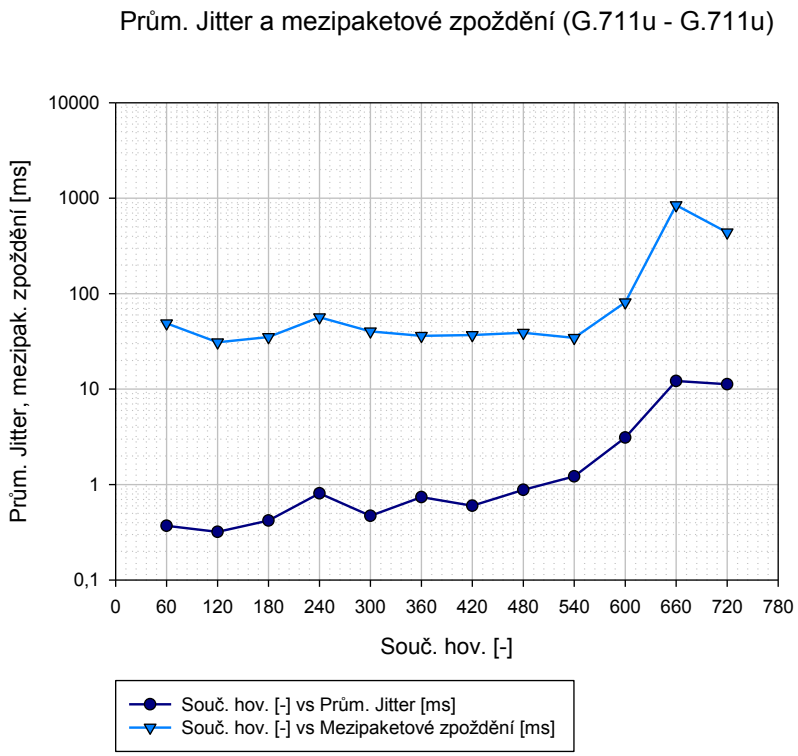
Obr. 8.6: Normalizované hodnoty vytížení CPU (G.711u – GSM)



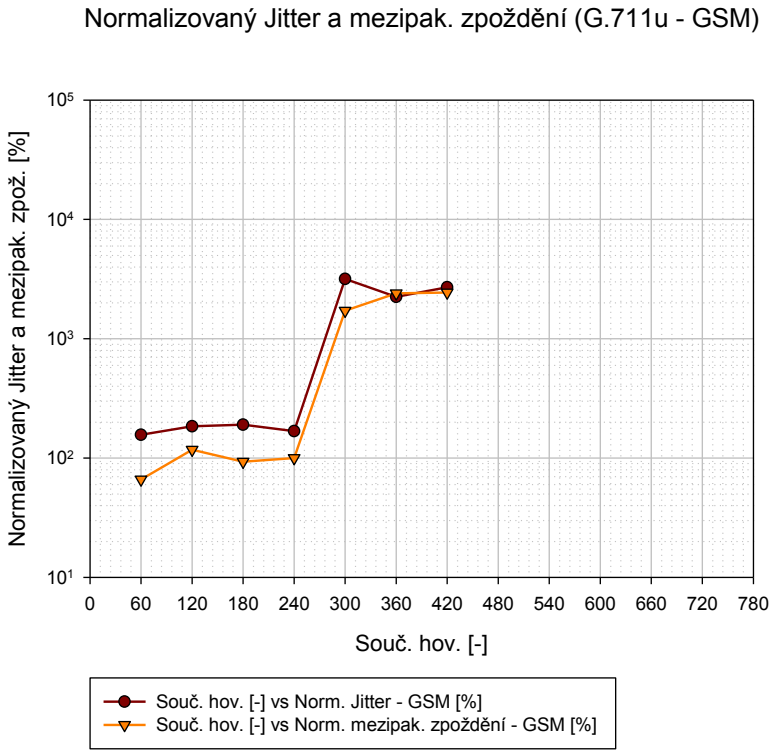
Obr. 8.7: RRD a SRD (G.711u – G.711u)



Obr. 8.8: Normalizované RRD a SRD (G.711u – GSM)



Obr. 8.9: Průměrný jitter a mezipaketové zpoždění (G.711u – G.711u)



Obr. 8.10: Normalizované hodnoty jitteru a mezipak. zpoždění (G.711u – GSM)

Pokles průměrného jitteru v oblasti s nízkou zátěží je podobně jako u RRD a SRD způsoben značnou variabilitou této měřené veličiny. V oblasti vysoké zátěže je pak na vině velké množství neúspěšných hovorů způsobené vysokou zátěží SIP serveru.

9 ZÁVĚR

Ačkoliv je problematika výkonnostního testování velmi obsáhlá, snažila se ji tato práce pojmout co nejobecněji. Od základních úvah o měření a jeho reprodukovatelnosti, přes vnesené chyby měření způsobené zejména nedokonalostí softwaru, které je třeba eliminovat správným časováním jednotlivých úloh, až po problematiku vyhodnocování výsledků. Obecný postup byl pak dále doplněn praktickou konfigurací testovací platformy na bázi open-source. Uvedený postup není jediným možným a ani jediným správným, to jakými prostředky bude testování prováděno, ovlivňuje konfiguraci testovací platformy, zejména její komplexnost a časovou náročnost, nemělo by však ovlivnit výsledky měření a jejich formát. Tato práce se snaží vycházet z připravovaného standardu, který je jednoznačným přínosem pro problematiku výkonnostního testování, a mnohé z jeho myšlenek jsou velmi dobře využitelné, jak ostatně ukázala i tato práce.

Vyhodnocení zásadních výsledků je věnována kapitola 8, vzhledem k rozsahu měřeni je úplný soubor vyhodnocení uveden v příloze. Pro jejich vyhodnocení může čtenář využít postupů uvedených v předchozí kapitole, přičemž pro jednoduchost lze vycházet z grafického zpracování těchto výsledků, které lze najít v tištěných přílohách. Z naměřených dat lze vyvodit závěry, které shrnuje následující tabulka:

	Maximální počet souč. hovorů podle kritéria			
Cílový kodek	CPU, paměť	RRD, SRD	IRA, neúsp. h.	Jitter, MPZ
G.711μ	660	**600**	660	**600**
G.711A	540	540	540	**480**
GSM	300	300	300	**240**
G.726-32b	**240**	**240**	**240**	**240**
	Maximální počet hovorů za sekundu podle kritéria			
Podprocesů	CPU, paměť	RRD, SRD	IRA, neúsp. h.	
Proxy - 4	2000	**1600**	**1600**	
Proxy - 16	2400	**1400**	1600	

Tab. 9.1: Shrnutí naměřených výsledků

Uvedené hodnoty jsou výsledkem vyhodnocení dat naměřených na testovací platformě využívající open source řešení. Testovací platforma byla realizována prostřednictvím série fyzických i virtuálních počítačů, které generovaly zátěž na protokolu SIP a RTP, kterou pak směrovaly vůči SIP serveru postaveném na low-end hardwaru (Athlon 64 X2, 4GB DDR2). V průběhu generování této zátěže pak byly měřeny parametry, uvedené v kategoriích tab. 9.1. Data, která byla měřením získána, byla graficky zpracována. Z grafických výsledků je pak možné vyčíst jednotlivé parametry uvedené tab. 9.1, jedná se totiž o limitní případy, kdy je

odečítána maximální hodnota současných hovorů (resp. hovorů za sekundu), při které ještě SIP server splňoval požadavky na kvalitu služby, které jsou definovány v kapitole 8.

Jelikož v textu není rozebrán případ měření na SIP Proxy, dovolím si zde malé doplnění. V uvedené tabulce jsou kategorie „Proxy – 4“ a „Proxy – 16“, toto číslo určuje, do kolika podprocesů (children) se Opensips podle nastavení mohl rozdělit. Můžeme tedy vyčíst, že 4 podprocesy podávaly lepší výkon než podprocesů 16, což je paradoxní, jelikož každý podproces je vlastně „naslouchač“ portu, tudíž při velkých vytíženích by měl Opensips excelovat při nastavení většího počtu podprocesů, avšak opak je pravdou.

Tab. 9.1 je tedy jednoduchým a čitelným výstupem této práce, jejíž dílčí výsledky jsou velmi komplexní a obsáhlé textové soubory vygenerované několika programy. Krom jiného můžeme zjistit, že nejnáročnějším z pohledu výkonnosti SIP serveru byl případ translace kodeků z G.711μ do G.726, i tak ale B2BUA byl schopen zpracovat 240 současných hovorů, což je pro běžné použití naprosto dostačující. Testovaný hardware, který se svým výkonem řadí do skupiny low-endových zařízení, tedy poskytuje dostatečný výkon i pro střední firmy, je třeba ale brát ohled na to, že ve skutečném firemním prostředí by dialplan Asterisku byl mnohem složitější, tudíž by i jeho provoz byl hardwarově náročnější. Naproti tomu výsledky SIP proxy jsou ze zcela jiné kategorie. Je zřejmé, že Opensips v dané konfiguraci je možné použít prakticky kdekoliv. Opět platí poznámka, že v reálném provozu by nastavení bylo složitější a tedy i požadovaný výkon vyšší, avšak rezervy této SIP proxy jsou obrovské. Navíc hardwarové vytížení bylo do značné míry způsobeno MySQL serverem, na kterém byly uloženy informace pro autentizaci, proto lze usuzovat, že použití jiné databáze by do značné míry zlepšilo možnosti tohoto SIP serveru. Bohužel, v době vzniku této práce nebylo možné zprovoznit databázi Berkeley, která je svou filosofií nejpodobnější interní databázi Asterisku, a to z důvodu nekompatibility databáze s novými utilitami, které databázi obsluhují.

Přínosem této práce je ucelená metodika pro testování SIP serverů v konfiguracích SIP Proxy i B2BUA. Výchozím bodem této práce jsou rozpracované drafty IETF a některé testy společnosti Transnexus, tato práce pak oba vstupy spojuje v jeden organický celek doplněný o zcela novou metodiku porovnávání výkonu B2BUA. Novinkou je také optimalizace testovací platformy, kdy je managementová role celého testovacího mechanismu integrována přímo do jednoho z klientů, čímž je možno odstranit z topologie redundantní počítač.

Autor této práce dlouhodobě spolupracuje se sdružením CESNET při řešení výzkumného záměru MŠMT - Multimediální přenosy a kolaborativní prostředí. Výkonnostní testování SIP infrastruktury je jedním z dílčích úkolů, na kterém autor ve sdružení CESNET pracuje již druhým rokem. Odsud pak autor získává stimuly pro další práci. Z praktického hlediska je do budoucna vhodné vyvinout nový přístup k výkonnostnímu testování ve formě komplexního a lépe škálovatelného nástroje s novou metodikou, který by v sobě integroval i modul pro zpracování výsledků. Z hlediska teoretického je pak nutné dotáhnout naznačenou metodiku do konečného stádia, které bude obsahovat komplexní a co nejobecnější postup pro výkonnostní testování. Na tomto je autor připraven pracovat během postgraduálního studia.

SEZNAM POUŽITÉ LITERATURY

[1] HLAVENKA, Jiří. *Částečně splněné naděje internetové telefonie*. Lupa [online]. 29. 12. 2009, [cit. 2010-03-18]. Dostupný z WWW: <http://www.lupa.cz/clanky/-castecne-splnene-nadeje-internetove-telefonie/>.

[2] *Asterisk - The open source telephony project* [online]. 2010 [cit. 2010-03-18]. About The Asterisk Project. Dostupné z WWW: <http://www.asterisk.org/asterisk>.

[3] *IxLoad Voice SIP* . [s.l.] : Ixia, 2009. 9 s. Dostupné z WWW: <http://ixia.org/pdfs/-datasheets/aptixia_ixload_sip.pdf>.

[4] *Methodology for Benchmarking SIP Networking Devices.* [s.l.] : IETF, 2010-02-08. 20 s. Dostupné z WWW: <http://tools.ietf.org/html/draft-ietf-bmwg-sip-bench-meth-01>.

[5] *Terminology for Benchmarking Session Initiation Protocol (SIP) Networking Devices.* [s.l.] : IETF, 2010-02-08. 34 s. Dostupné z WWW: <http://tools.ietf.org/html/draft-ietf-bmwg-sip-bench-term-01>.

[6] *SIP End-to-End Performance Metrics.* [s.l.] : IETF, 2009-09-09. 32 s. Dostupné z WWW: <http://tools.ietf.org/html/draft-ietf-pmol-sip-perf-metrics-04>.

[7] *Transnexus - Least Cost Routing, Number Portability, Traffic Reports, Profit Analysis and Wholesale Billing for VoIP Networks* [online]. 2006 [cit. 2010-03-23]. Dostupné z WWW: <http://www.transnexus.com/>.

[8] *Performance Benchmark Test for Asterisk B2BUA.* [s.l.] : Transnexus, 2008-10-03. 66 s. Dostupné z WWW: <http://www.transnexus.com/White%20Papers/Asterisk_V-1.4_Benchmark_Test_2008-10-21.pdf>.

[9] *Performance Benchmark Test for OpenSER and SIP Express Router*. [s.l.] : Transnexus, 2007-06-04. 41 s. Dostupné z WWW: <http://www.transnexus.com/-White%20Papers/2007-06-04_OpenSER-SER_Benchmark_Test.pdf>.

[10] *SIPp* [online]. 2009 [cit. 2010-04-01]. IMS Bench SIPp. Dostupné z WWW: <http://sipp.sourceforge.net/ims_bench_sipp/index.html>.

[11] *SIPp* [online]. 2009 [cit. 2010-04-01]. SIPp reference documentation. Dostupné z WWW: <http://sipp.sourceforge.net/doc3.0/reference.html#Compiling+SIPp>.

[12] *SYSSTAT* [online]. 2009 [cit. 2010-04-02]. SAR manual page. Dostupné z WWW: <http://pagesperso-orange.fr/sebastien.godard/man_sar.html>.

[13] *Telegro.cz* [online]. 2009 [cit. 2010-04-02]. Konfigurace Asterisku (2) - Konfigurační soubory. Dostupné z WWW: <http://www.telegro.cz/2009/03/17/konfigurace-asterisku-2-konfiguracni-soubory>.

[14] *OSnews.com* [online]. 2009 [cit. 2010-04-02]. Awk and Sed One-Liners Explained. Dostupné z WWW: <http://www.osnews.com/story/21004/Awk_and_Sed_One-Liners_Explained>.

[15] MILLER, Mark. *Understanding H.323—Part I: History and Architecture*. VoIPPlanet [online]. 2005-04-19, [cit. 2010-04-18]. Dostupný z WWW: <http://-www.voipplanet.com/backgrounders/article.php/3498736>.

[16] VOZŇÁK, Miroslav. *Voice over IP*. Ostrava: VŠB-TUO, 2008. Prvky H.323 a jejich vlastnosti, s. 51-52. ISBN 978-80-248-1828-3.

[17] VOZŇÁK, Miroslav. *Voice over IP*. Ostrava: VŠB-TUO, 2008. Základní popis protokolu SIP, s. 91-93. ISBN 978-80-248-1828-3.

[18] VOZŇÁK, Miroslav. *Voice over IP*. Ostrava: VŠB-TUO, 2008. Dialog, s. 106-107. ISBN 978-80-248-1828-3.

[19] G.711 In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-04-18]. Dostupné z WWW: <http://-en.wikipedia.org/wiki/G.711>.

[20] IXIA CHANGES THE ECONOMICS OF SERVICE VERIFICATION WITH CENTRALIZED SOLUTION FOR SERVICE PROVIDERS. *Ixia Newsletter* [online]. 2009-03-24, [cit. 2010-04-26]. Dostupný z WWW: <http://info.ixiacom.com/IxiaCustomerNewsletter0409.html>.

SEZNAM PŘÍLOH

TIŠTĚNÉ PŘÍLOHY

ELEKTRONICKÉ PŘÍLOHY